PRACTICAL NO : 01

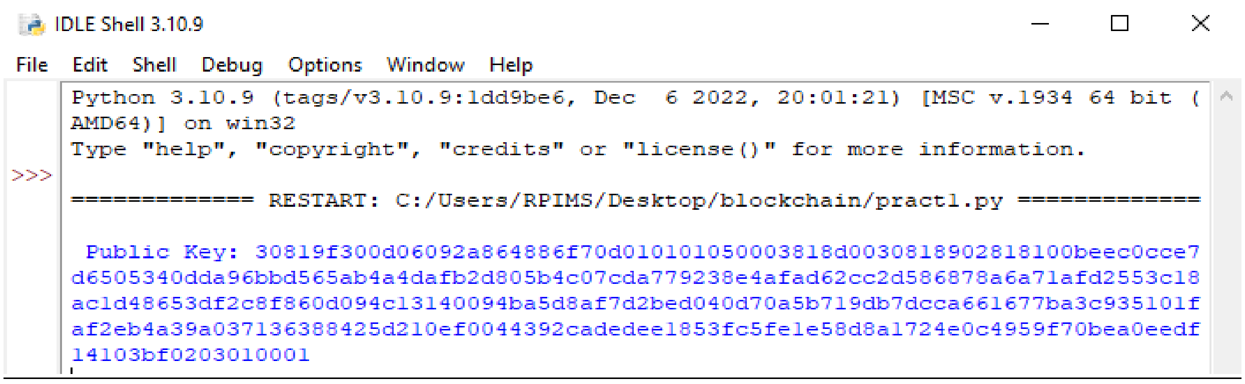
Write the following programs for blockchain in python

A) Aim : A simple client class that generates the private and public keys by using the built- in Python RSA algorithm and test it.

STEPS:
Setup the Environment Variable to appropriate path Install Pycryptodome using “pip intsall pycryptodome”

```
import binascii
import Crypto
from Crypto.PublicKey import RSA
from Crypto.Signature import PKCS1_v1_5
class Client:
   def __init__(self):
      random = Crypto.Random.new().read
      self._private_key = RSA.generate(1024, random)
      self._public_key = self._private_key.publickey()
      self._signer = PKCS1_v1_5.new(self._private_key)
   @property
   def identity(self):
      return binascii.hexlify(self._public_key.exportKey(format="DER")).decode("ascii")

Srinivas = Client()
print("\n Public Key:",Srinivas.identity)
```

Output :



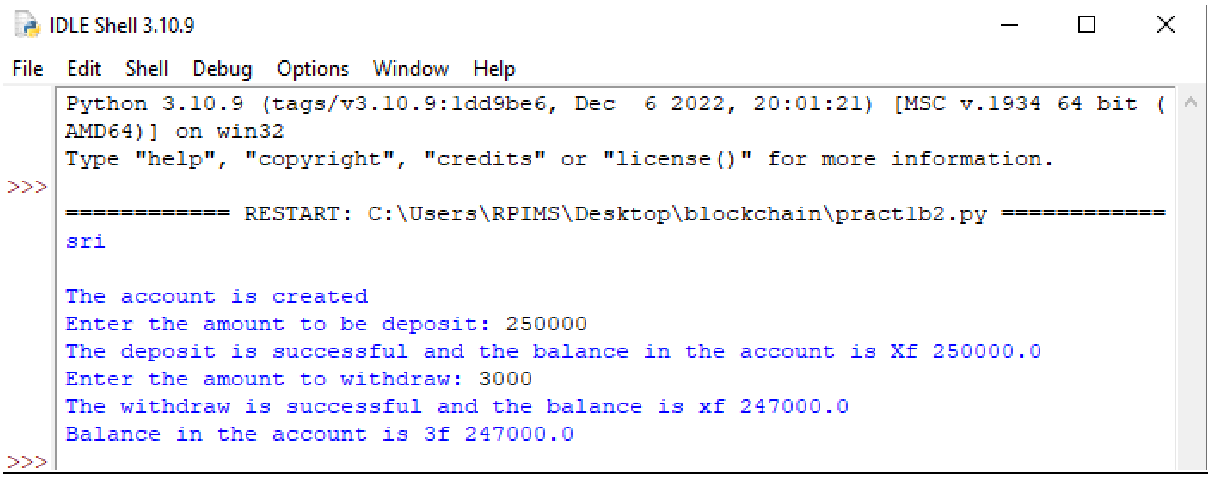
```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:/Users/RPIMS/Desktop/blockchain/pract1.py =============

 Public Key: 30819f300d06092a864886f70d010101050003818d0030818902818100beec0cce7
d6505340dda96bbd565ab4a4dafb2d805b4c07cda779238e4afad62cc2d586878a6a71afd2553c18
ac1d48653df2c8f860d094c13140094ba5d8af7d2bed040d70a5b719db7dcca661677ba3c935101f
af2eb4a39a037136388425d210ef0044392cadedee1853fc5fe1e58d8a1724e0c4959f70bea0eedf
14103bf0203010001
```

B) A transaction class to send and receive money and test it.

Steps: Install “client” module

```
class Bank:
  def __init__(self):
    self.balance = 0
    print("sri \n")
    print("The account is created")

  def deposit(self):
    amount = float(input("Enter the amount to be deposit: "))
    self.balance = self.balance + amount
    print ("The deposit is successful and the balance in the account is Xf", self.balance)

  def withdraw(self):
    amount = float(input("Enter the amount to withdraw: "))
    if (self.balance >= amount):
      self.balance = self.balance - amount
      print ("The withdraw is successful and the balance is xf", self.balance)
    else:
      print("Insuficient Balance")

  def enquiry(self):
    print ("Balance in the account is 3f",self.balance)

acc = Bank()
acc.deposit()
acc.withdraw()
acc.enquiry()
```

Output :

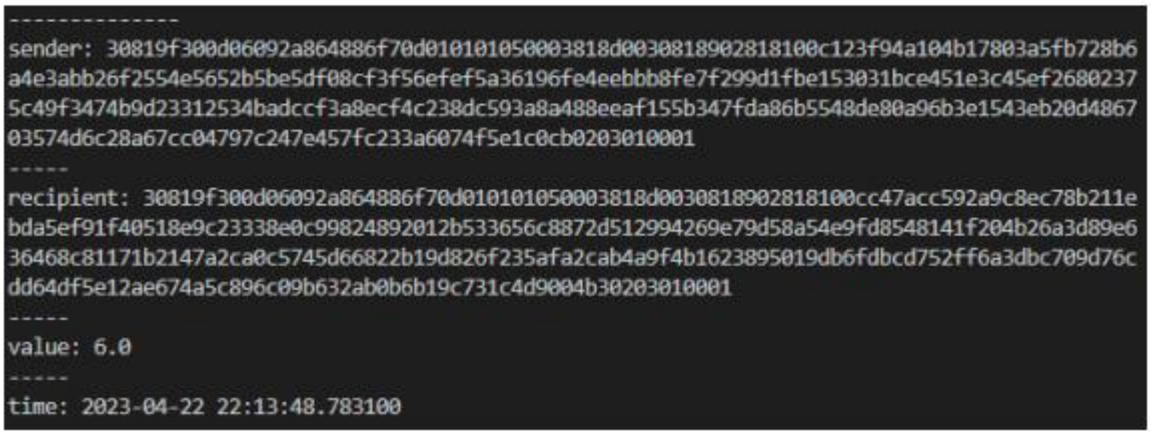


```
Python 3.10.9 (tags/v3.10.9:1dd9be6, Dec  6 2022, 20:01:21) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\RPIMS\Desktop\blockchain\pract1b2.py ============
sri

The account is created
Enter the amount to be deposit: 250000
The deposit is successful and the balance in the account is Xf 250000.0
Enter the amount to withdraw: 3000
The withdraw is successful and the balance is xf 247000.0
Balance in the account is 3f 247000.0
>>>
```

C) Create multiple transactions and display them.

```
from client import client
from transaction_class import transaction
Dinesh = client()
Ramesh = client()
t = Transaction(Dinesh, Ramesh.identity, 5.0)
print("\nTransaction Recipient:\n", t.recipient)
# print("\nTransaction Sender:\n", t.sender)
print("\nTransaction Value:\n", t.value)

signature = t.sign_transaction()
print("\nSignature:\n", signature)

Dinesh = Client()
Ramesh = Client()
Seema = Client()
Vijay = Client()
t1 = Transaction(Dinesh, Ramesh.identity, 15.0)
t1.sign_transaction()
transactions = [t1]
t2 = Transaction(Dinesh, Seema.identity, 6.0)
t2.sign_transaction()
transactions.append(t2)
t3 = Transaction(Ramesh, Vijay.identity, 2.0)
t3.sign_transaction()
transactions.append(t3)
t4 = Transaction(Seema, Ramesh.identity, 4.0)
t4.sign_transaction()
transactions.append(t4)

for transaction in transactions:
   Transaction.display_transaction(transaction)
   print("--------------")
```

Output :

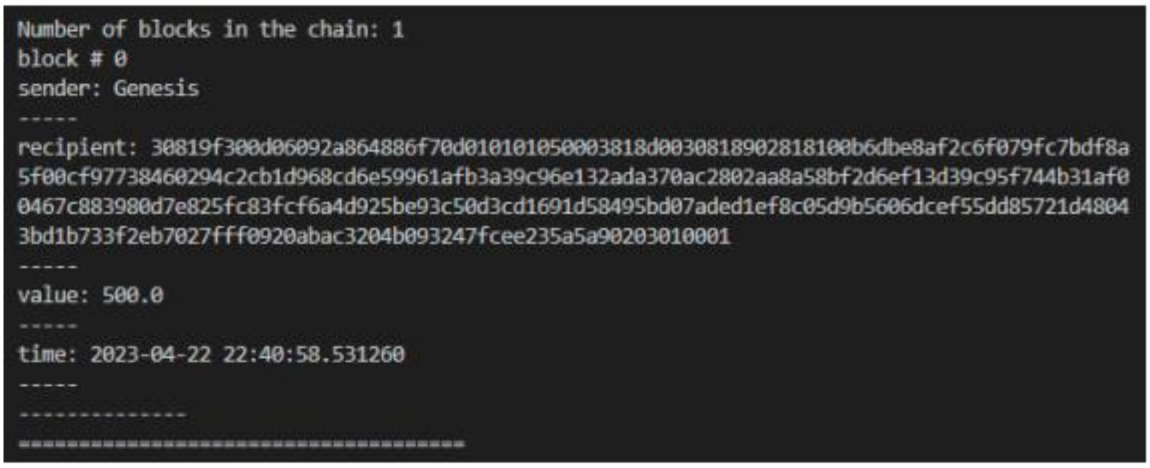


```
--------------
sender: 30819f300d06092a864886f70d010101050003818d0030818902818100c123f94a104b17803a5fb728b6
a4e3abb26f2554e5652b5be5df08cf3f56efef5a36196fe4eebbb8fe7f299d1fbe153031bce451e3c45ef2680237
5c49f3474b9d23312534badccf3a8ecf4c238dc593a8a488eeaf155b347fda86b5548de80a96b3e1543eb20d4867
03574d6c28a67cc04797c247e457fc233a6074f5e1c0cb0203010001
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100cc47acc592a9c8ec78b211e
bda5ef91f40518e9c23338e0c99824892012b533656c8872d512994269e79d58a54e9fd8548141f204b26a3d89e6
36468c81171b2147a2ca0c5745d66822b19d826f235afa2cab4a9f4b1623895019db6fdbcd752ff6a3dbc709d76c
dd64df5e12ae674a5c896c09b632ab0b6b19c731c4d9004b30203010001
-----
value: 6.0
-----
time: 2023-04-22 22:13:48.783100
```

D) Create a blockchain, a genesis block and execute it.

```
from client import Client
from transaction_class import Transaction

class Block:
    def __init__(self, client):
        self.verified_transactions = []
        self.previous_block_hash = ""
        self.Nonce = ""
        self.client = client

def dump_blockchain(blocks):
    print(f"\nNumber of blocks in the chain: {len(blocks)}")
    for i, block in enumerate(blocks):
        print(f"block # {i}")
            for transaction in block.verified_transactions:
                Transaction.display_transaction(transaction)
                print("--------------")
print("====================================")
Dinesh = Client()
t0 = Transaction("Genesis", Dinesh.identity(), 500.0)
block0 = Block(Dinesh)
block0.previous_block_hash = ""
NONCE = None
block0.verified_transactions.append(t0)
digest = hash(block0)
last_block_hash = digest
TPCoins = [block0]
dump_blockchain(TPCoins)
```

<u>Output</u> :

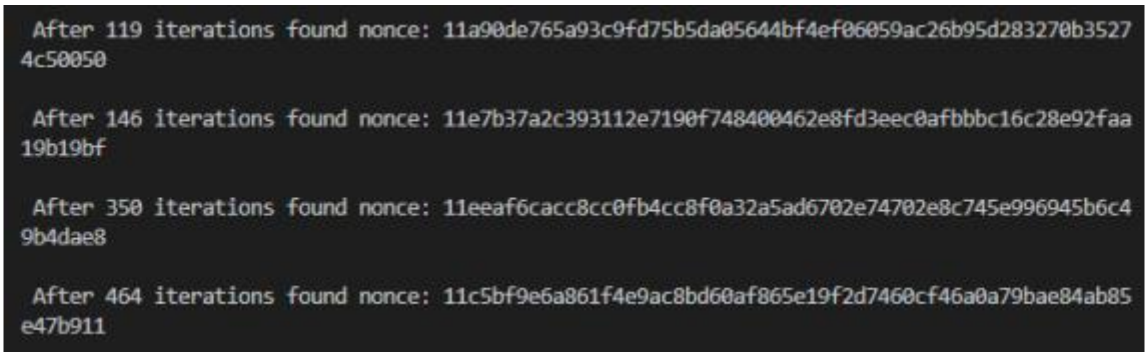


```
Number of blocks in the chain: 1
block # 0
sender: Genesis
-----
recipient: 30819f300d06092a864886f70d010101050003818d0030818902818100b6dbe8af2c6f079fc7bdf8a
5f00cf97738460294c2cb1d968cd6e59961afb3a39c96e132ada370ac2802aa8a58bf2d6ef13d39c95f744b31af0
0467c883980d7e825fc83fcf6a4d925be93c50d3cd1691d58495bd07aded1ef8c05d9b5606dcef55dd85721d4804
3bd1b733f2eb7027fff0920abac3204b093247fcee235a5a90203010001
-----
value: 500.0
-----
time: 2023-04-22 22:40:58.531260
-----
--------------
==================================
```

E) Create a mining function and test it.

```
import hashlib
def sha256(message):
    return hashlib.sha256(message.encode("ascii")).hexdigest()
def mine(message, difficulty=1):
        assert difficulty = 1
        prefix = "1" * difficulty
        for i in range(1000):
                digest = sha256(str(hash(message)) + str(i))
                 If digest.startswith(prefix):
                        print(f"after {str(i)} iterations found nonce: {digest}")
                        # return print(digest)
mine("test message", 2)
```

Output :

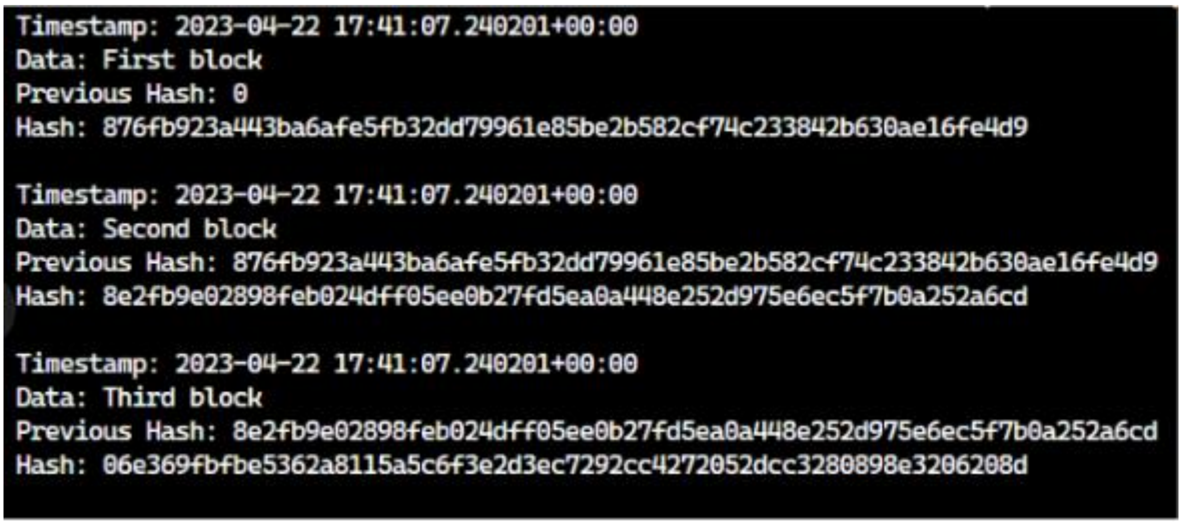


```
 After 119 iterations found nonce: 11a90de765a93c9fd75b5da05644bf4ef06059ac26b95d283270b3527
4c50050

 After 146 iterations found nonce: 11e7b37a2c393112e7190f748400462e8fd3eec0afbbbc16c28e92faa
19b19bf

 After 350 iterations found nonce: 11eeaf6cacc8cc0fb4cc8f0a32a5ad6702e74702e8c745e996945b6c4
9b4dae8

 After 464 iterations found nonce: 11c5bf9e6a861f4e9ac8bd60af865e19f2d7460cf46a0a79bae84ab85
e47b911
```

F) Add blocks to the miner and dump the blockchain.

```
import datetime
import hashlib

# Create a class with two functions
class Block:
    def __init__(self, data, previous_hash):
        self.timestamp = datetime.datetime.now(datetime.timezone.utc)
        self.data = data
        self.previous_hash = previous_hash
        self.hash = self.calc_hash()
    def calc_hash(self):
        sha = hashlib.sha256()
        hash_str = self.data.encode("utf-8")
        sha.update(hash_str)
        return sha.hexdigest()

# Instantiate the class
blockchain = [Block("First block", "0")]
blockchain.append(Block("Second block", blockchain[0].hash))
blockchain.append(Block("Third block", blockchain[1].hash))
# Dumping the blockchain
for block in blockchain:
        print(f"Timestamp: {block.timestamp}\nData: {block.data}\nPrevious
        Hash:{block.previous_hash}\nHash: {block.hash}\n")
```

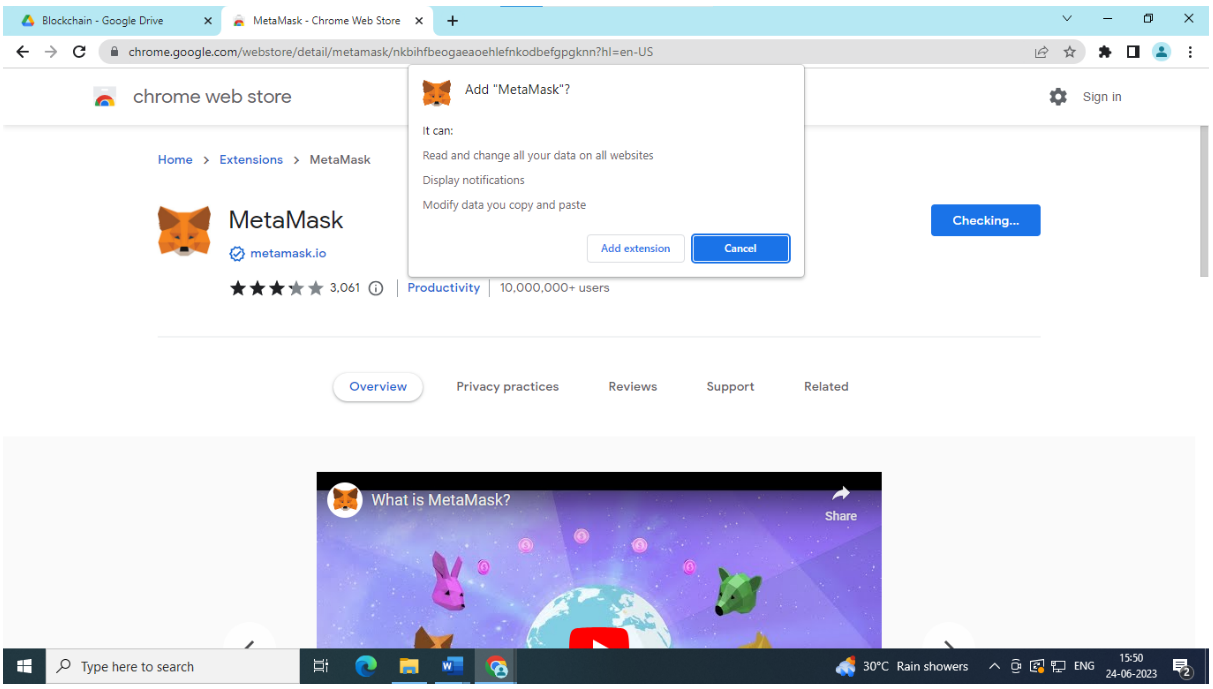
<u>Output</u> :



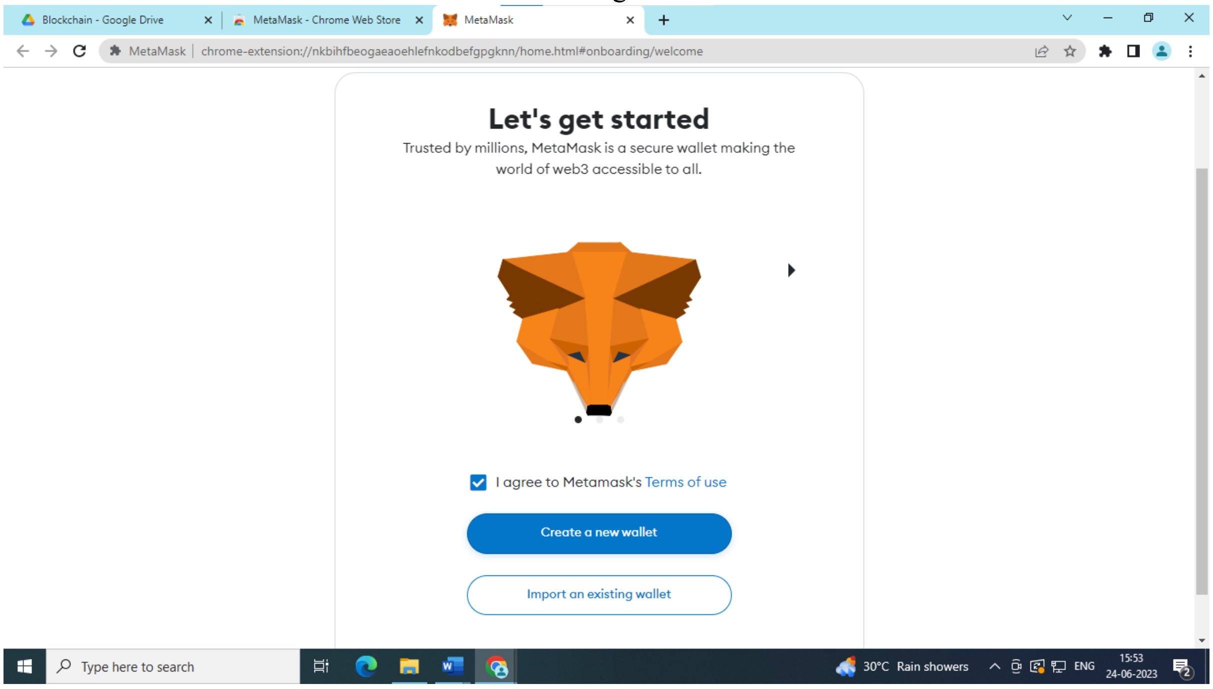
```
Timestamp: 2023-04-22 17:41:07.240201+00:00
Data: First block
Previous Hash: 0
Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9

Timestamp: 2023-04-22 17:41:07.240201+00:00
Data: Second block
Previous Hash: 876fb923a443ba6afe5fb32dd79961e85be2b582cf74c233842b630ae16fe4d9
Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd

Timestamp: 2023-04-22 17:41:07.240201+00:00
Data: Third block
Previous Hash: 8e2fb9e02898feb024dff05ee0b27fd5ea0a448e252d975e6ec5f7b0a252a6cd
Hash: 06e369fbfbe5362a8115a5c6f3e2d3ec7292cc4272052dcc3280898e3206208d
```

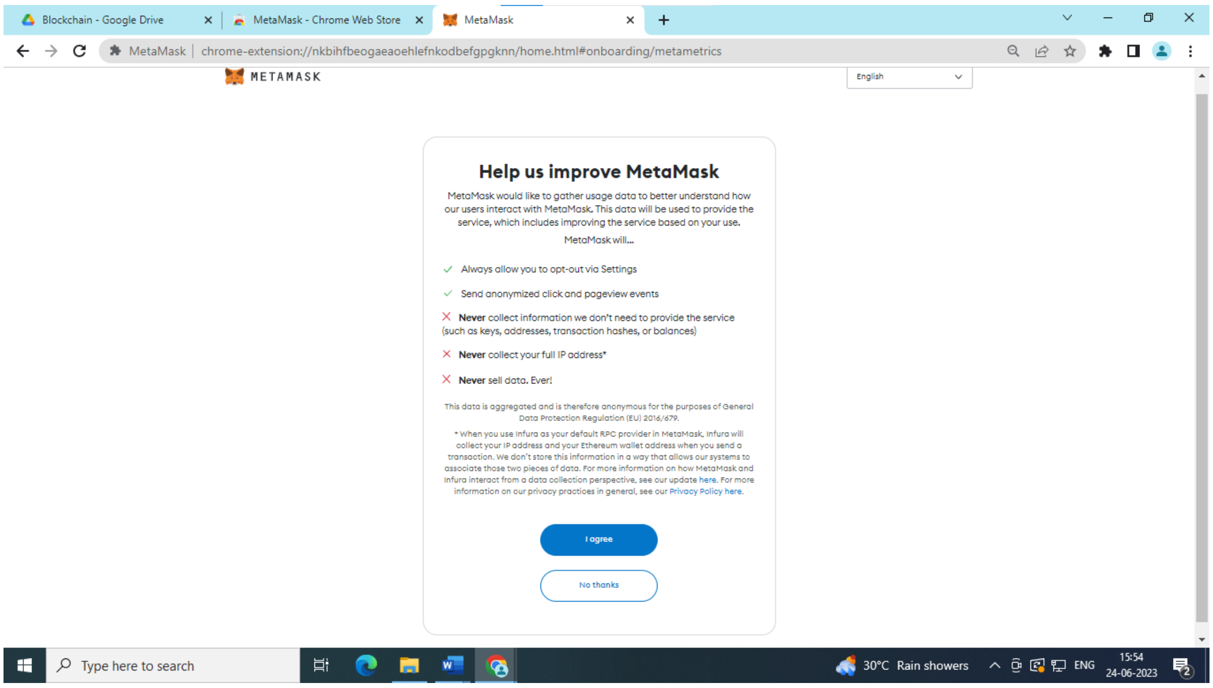
PRACTICAL NO : 02
Install and configure Go Ethereum and the Mist browser. Develop and test a sample application.(MetaMask & Remix)

Step 1 -> Install MetaMask extension for chrome from Chrome Web Store



Step 2-> Click on Metamask Extension in Extensions. Below page will open in a new tab. Click on Create a New Wallet. Click on I agree.


Let's get started
Trusted by millions, MetaMask is a secure wallet making the world of web3 accessible to all.
I agree to Metamask's Terms of use
Create a new wallet
Import an existing wallet


Blockchain - Google Drive
MetaMask - Chrome Web Store
MetaMask
MetaMask | chrome-extension://nkbihfbeogaeaoehlefnkodbefgpgknn/home.html#onboarding/metametrics
METAMASK
English
Type here to search
30°C Rain showers
ENG
15:54
24-06-2023

Help us improve MetaMask

MetaMask would like to gather usage data to better understand how our users interact with MetaMask. This data will be used to provide the service, which includes improving the service based on your use.

MetaMask will...

✓ Always allow you to opt-out via Settings

✓ Send anonymized click and pageview events

✗ **Never** collect information we don't need to provide the service (such as keys, addresses, transaction hashes, or balances)

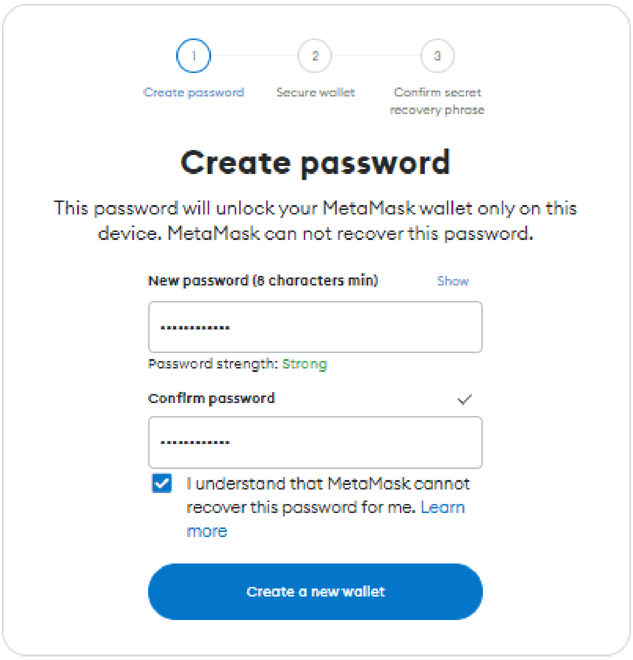
✗ **Never** collect your full IP address*

✗ **Never** sell data. Ever!

This data is aggregated and is therefore anonymous for the purposes of General Data Protection Regulation (EU) 2016/679.

* When you use Infura as your default RPC provider in MetaMask, Infura will collect your IP address and your Ethereum wallet address when you send a transaction. We don't store this information in a way that allows our systems to associate those two pieces of data. For more information on how MetaMask and Infura interact from a data collection perspective, see our update here. For more information on our privacy practices in general, see our Privacy Policy here.

Step 3-> Create a password. This password can be used only on the device it was created on. Create a Strong password and click on Create a new Wallet button



Create password

This password will unlock your MetaMask wallet only on this device. MetaMask can not recover this password.

New password (8 characters min) Show

•••••••••••••

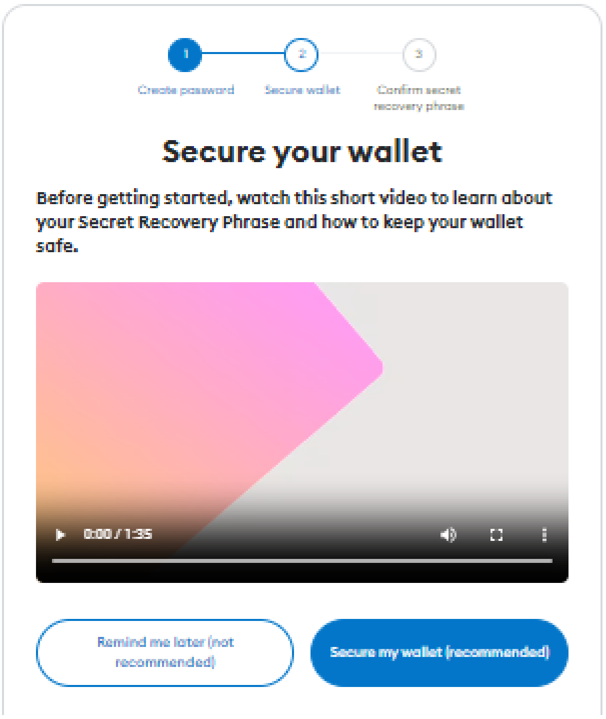
Password strength: Strong

Confirm password ✓

•••••••••••••

☑ I understand that MetaMask cannot recover this password for me. Learn more

Step 4-> Click on Secure my wallet button, following window will appear

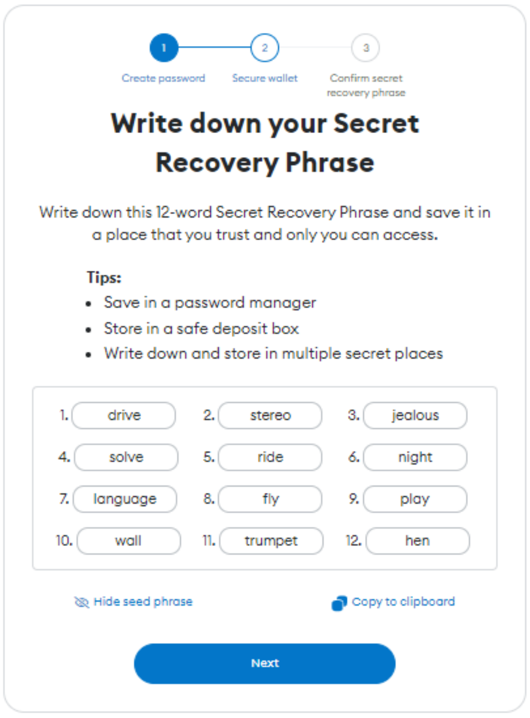

1 Create password
2 Secure wallet
3 Confirm secret recovery phrase

Secure your wallet

Before getting started, watch this short video to learn about your Secret Recovery Phrase and how to keep your wallet safe.

0:00 / 1:35
Remind me later (not recommended)
Secure my wallet (recommended)

Step 5-> Click on Reveal Secret Recovery Phrase button and save the words in the same sequence



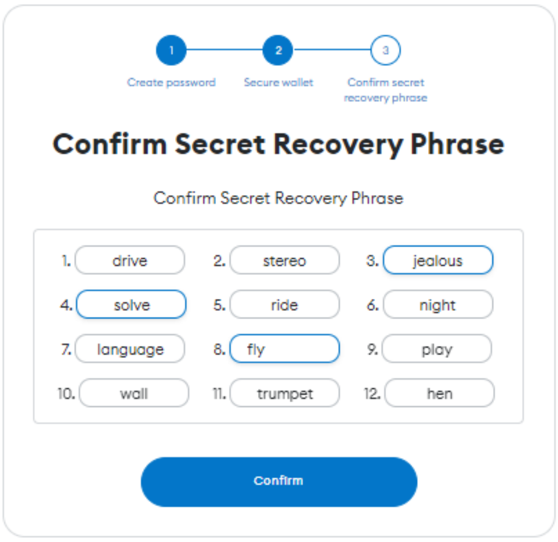
Write down your Secret Recovery Phrase

Write down this 12-word Secret Recovery Phrase and save it in a place that you trust and only you can access.

Tips:
- Save in a password manager
- Store in a safe deposit box
- Write down and store in multiple secret places

1. drive	2. stereo	3. jealous
4. solve	5. ride	6. night
7. language	8. fly	9. play
10. wall	11. trumpet	12. hen

Step 6-> Enter the respective words in the empty positions and click Confirm.

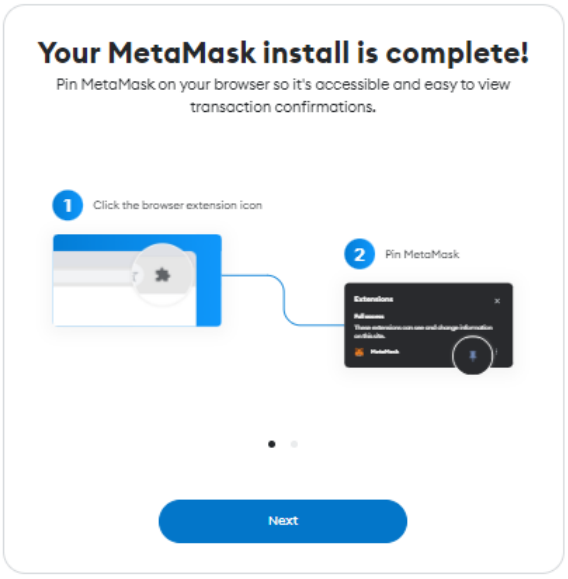


Confirm Secret Recovery Phrase

Confirm Secret Recovery Phrase

1. drive	2. stereo	3. jealous
4. solve	5. ride	6. night
7. language	8. fly	9. play
10. wall	11. trumpet	12. hen

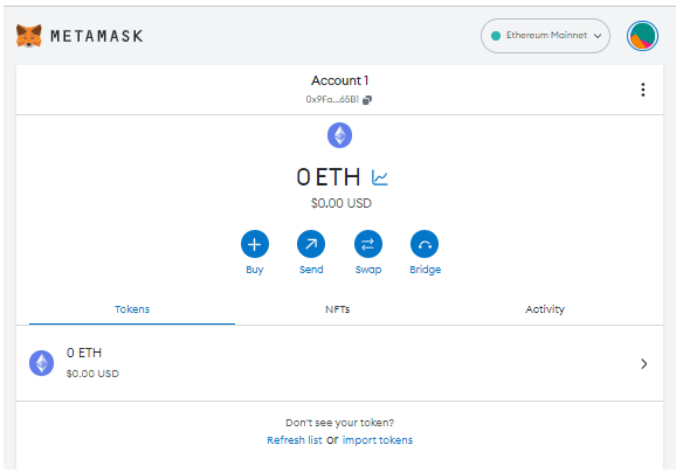
Step 7-> Click Got it!
Step 8-> Click on Next



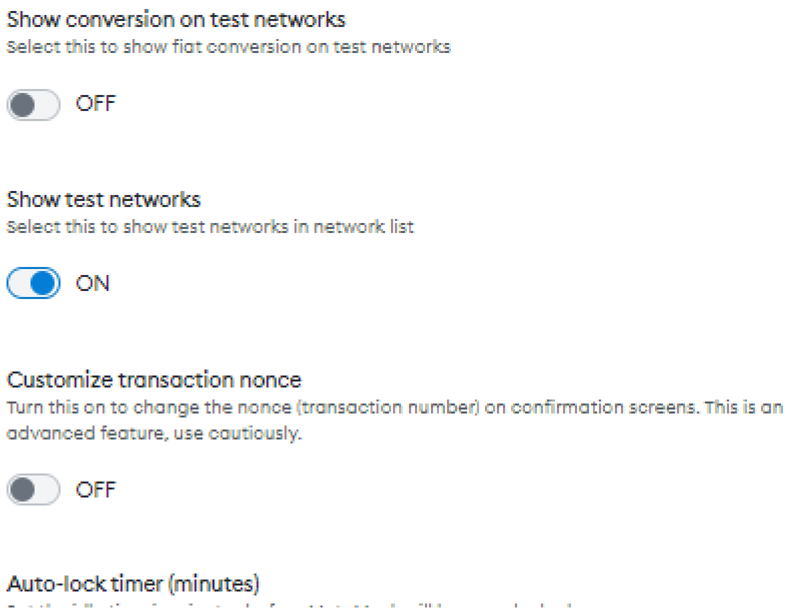
Your MetaMask install is complete!

Pin MetaMask on your browser so it's accessible and easy to view transaction confirmations.

Step 9-> Following will be the Dashboard



Step 10-> Click on Ethereum Mainnet button. Next click on Show/hide test networks.



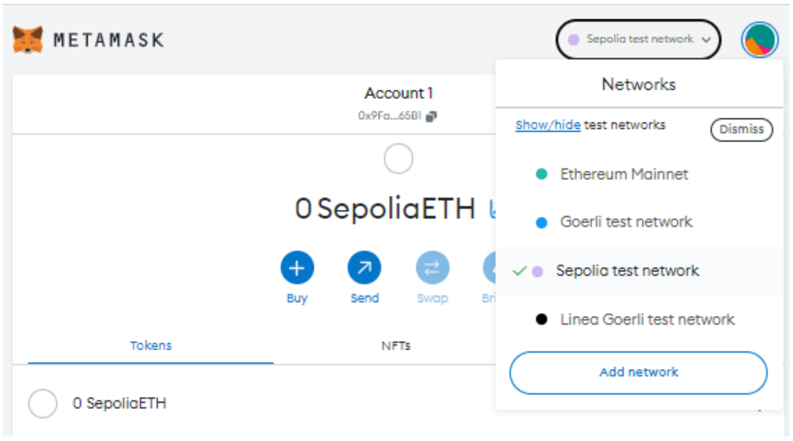
Show conversion on test networks
Select this to show fiat conversion on test networks
OFF

Show test networks
Select this to show test networks in network list
ON

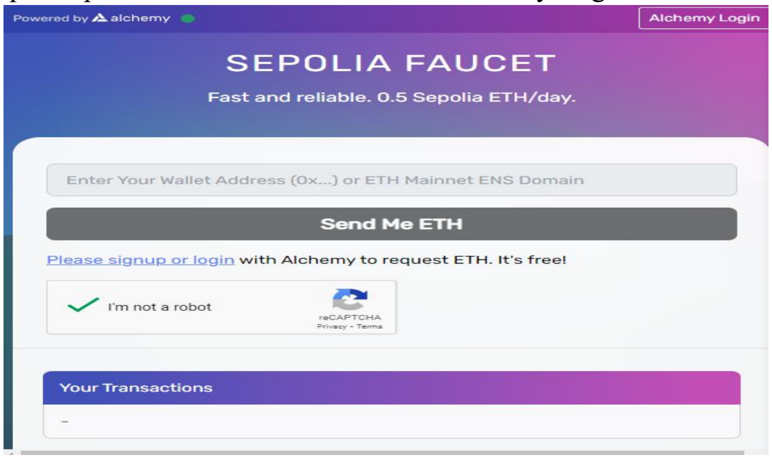
Customize transaction nonce
Turn this on to change the nonce (transaction number) on confirmation screens. This is an advanced feature, use cautiously.
OFF

Auto-lock timer (minutes)

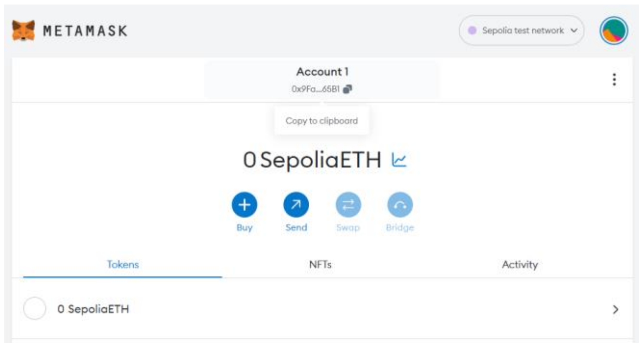
Step 11-> Check if tesnets are shown by clicking on Etherum Mainnet button. Click on Sepolia test network.



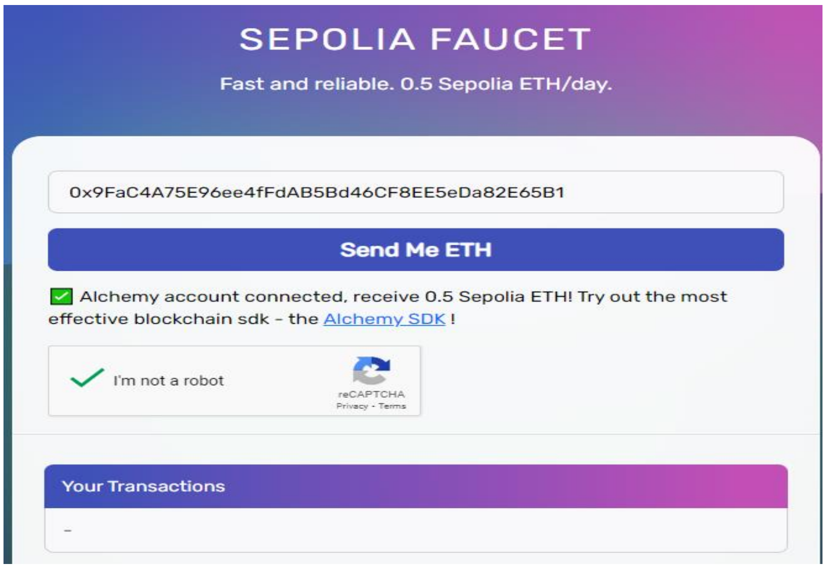
Step 12-> Go to https://sepoliafaucet.com/ and Click on Alchemy Login button.


Powered by alchemy
Alchemy Login
SEPOLIA FAUCET
Fast and reliable. 0.5 Sepolia ETH/day.
Enter Your Wallet Address (0x...) or ETH Mainnet ENS Domain
Send Me ETH
Please signup or login with Alchemy to request ETH. It's free!
I'm not a robot
reCAPTCHA
Privacy - Terms
Your Transactions

Step 13-> Login to a gmail account in another browser tab and click on Sign in with Google
Step 14-> Now go to MetaMask and copy the account address.



Step 15-> Paste the address and click on Send Me ETH.


SEPOLIA FAUCET

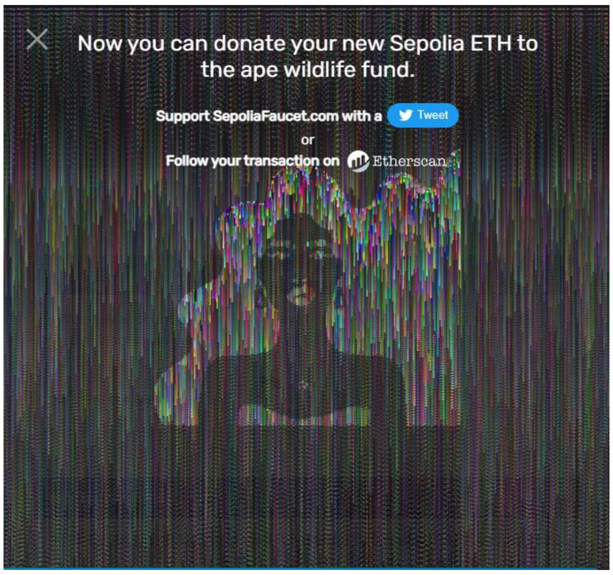
Fast and reliable. 0.5 Sepolia ETH/day.

0x9FaC4A75E96ee4fFdAB5Bd46CF8EE5eDa82E65B1

Send Me ETH

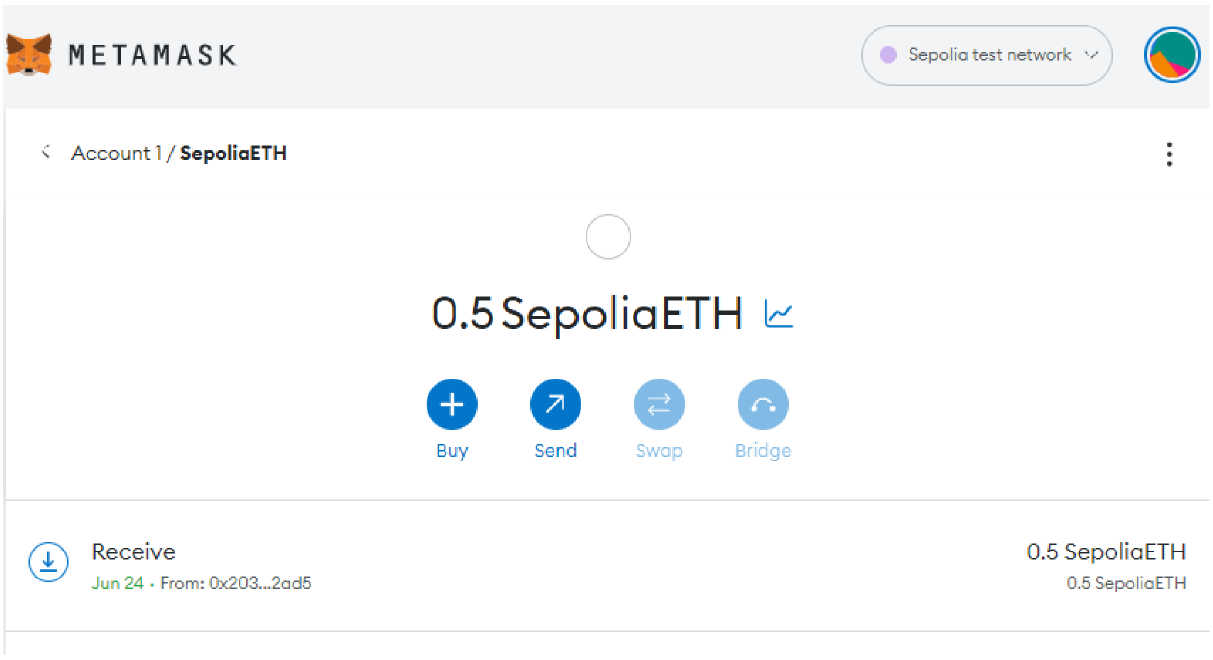
Alchemy account connected, receive 0.5 Sepolia ETH! Try out the most effective blockchain sdk - the Alchemy SDK !

Step 16-> Your ETH transfer is succesfull. You should see a similar animation.


Now you can donate your new Sepolia ETH to the ape wildlife fund.

Support SepoliaFaucet.com with a Tweet
or
Follow your transaction on Etherscan

Step 17-> Check your MetaMask account for Sepolia test network. 0.5 ETH will be added.



Account 1 / **SepoliaETH**

0.5 SepoliaETH

Receive
Jun 24 · From: 0x203...2ad5
0.5 SepoliaETH
0.5 SepoliaETH

PRACTICAL NO : 03

Implement and demonstrate the use of the following in solidity

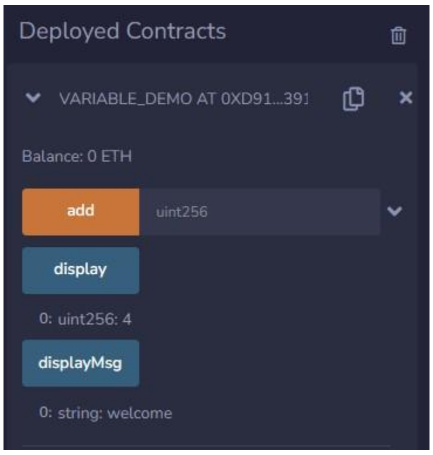
STEPS:
1. To execute solidity scripts go to ->https://remix.ethereum.org/
2. Open contracts folder and starting writing scripts. The scripts are compiled using solidity compiler.
3. The following scripts were compiled using 0.5.0+commit.1d4f565a solidity compiler
4. Deploy the scripts to execute code

A) Aim : Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables

1. Variables

```
pragma solidity ^0.5.0;
contract variable_demo {
        uint256 sum = 4; //state variable
        uint256 x;
        address a;
        string s = "welcome";
function add(uint256) public {
        uint256 y = 2; //local variable sum = sum+x+y:
        sum = sum + x + y;
}
function display() public view returns (uint256) {
        return sum;
        }
function displayMsg() public view returns (string memory) {
        return s;
    }
}
```

Output :

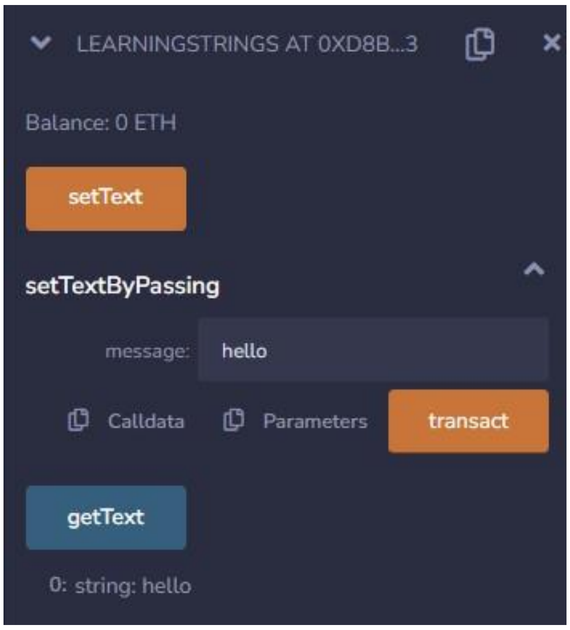


```
Balance: 0 ETH
0: uint256: 4
0: string: welcome
```

2. Strings

```
pragma solidity ^0.5.0;
contract LearningStrings {
        string text;
     function getText() public view returns (string memory) {
             return text;
             }
       function setText() public {
                 text = "hello";
             }
        function setTextByPassing(string memory message) public {
             text = message;
   }
}
```

Output :



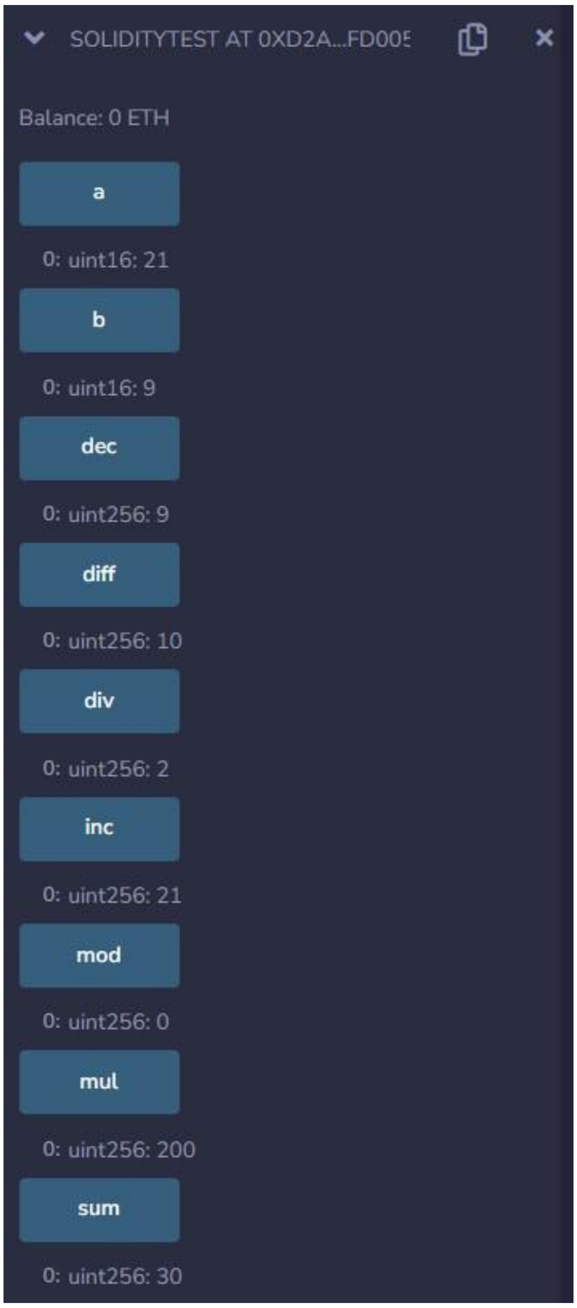
```
Balance: 0 ETH
message: hello
0: string: hello
```

3. Operators

```
pragma solidity ^0.5.0;
contract SolidityTest {
             uint16 public a = 20;
             uint16 public b = 10;
             uint256 public sum = a + b;
             uint256 public diff = a - b;
             uint256 public mul = a * b;
             uint256 public div = a / b;
             uint256 public mod = a % b;
             uint256 public dec = --b;
             uint256 public inc = ++a;
```

}

Output :

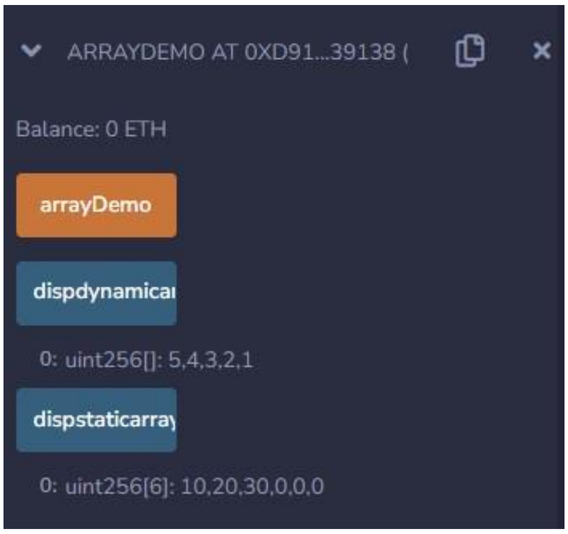


```
Balance: 0 ETH
0: uint16: 21
0: uint16: 9
0: uint256: 9
0: uint256: 10
0: uint256: 2
0: uint256: 21
0: uint256: 0
0: uint256: 200
0: uint256: 30
```

4. Array

```
pragma solidity ^0.5.0;
contract arraydemo
{
   //Static Array
    uint[6] arr2=[10,20,30];
     function dispstaticarray() public view returns(uint[6] memory)
    {
     return arr2;
      }
//Dynamic Array
uint x=5;
uint [] arr1;
function arrayDemo() public
{
            while(x>0)
            {
             arr1.push(x);
              x=x-1;
            }
}
function dispdynamicarray() public view returns(uint[] memory)
   {
      return arr1;
   }
}
```

Output :



```
ARRAYDEMO AT 0XD91...39138 (
Balance: 0 ETH
arrayDemo
dispdynamicai
0: uint256[]: 5,4,3,2,1
dispstaticarray
0: uint256[6]: 10,20,30,0,0,0
```

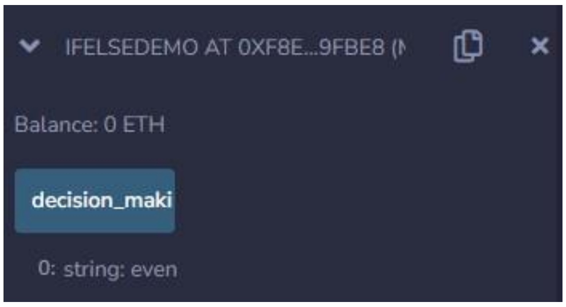
5. Decision Making

If Else

```
pragma solidity ^0.5.0;
contract ifelsedemo
{
            uint i=10;
            function decision_making() public view returns(string memory)
            {
```

```
            if(i%2==0)
        {
                return "even";
        }
        else
        {
                return "Odd";
            }
        }
}
```

Output :



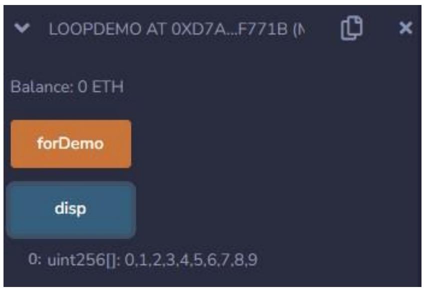
```
IFELSEDEMO AT 0XF8E...9FBE8 (M
Balance: 0 ETH
decision_maki
0: string: even
```

6. Loops

For Loop

```
pragma solidity ^0.5.0;
contract loopDemo
{
        uint [] data;
        function forDemo() public returns(uint[] memory)
        {
        for(uint i=0; i<10; i++){
            data.push(i);
        }
        return data;
}
function disp() public view returns(uint[] memory)
{
    return data;
  }
}
```

Output :

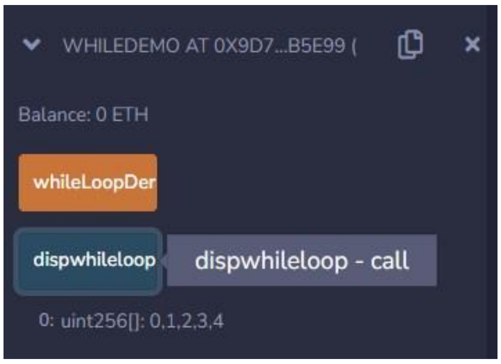


```
LOOPDEMO AT 0XD7A...F771B (M
Balance: 0 ETH
forDemo
disp
0: uint256[]: 0,1,2,3,4,5,6,7,8,9
```

While Loop

```
pragma solidity ^0.5.0;
contract whiledemo
{
            uint [] data;
            uint x=0;

function whileLoopDemo() public
{
   while(x<5)
{
            data.push(x);
            x=x+1;
}
}
function dispwhileloop() public view returns(uint[] memory)
 {
            return data;
             }
}
```

Output :

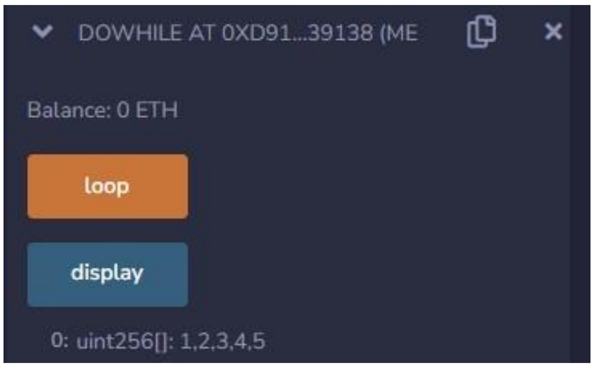


```
WHILEDEMO AT 0X9D7...B5E99 (
Balance: 0 ETH
0: uint256[]: 0,1,2,3,4
```

Do While

```
pragma solidity ^0.5.0;
// Creating a contract
contract DoWhile {
        // Declaring a dynamic array
            uint256[] data;
       // Declaring state variable
            uint8 j = 0;
        // Defining function to demonstrate
            // 'Do-While loop'
function loop() public returns (uint256[] memory) {
            do {
                        j++;
                        data.push(j);
                } while (j < 5);
                return data;
}
            function display() public view returns(uint256[] memory){
            return data;
            }
}
```

Output :

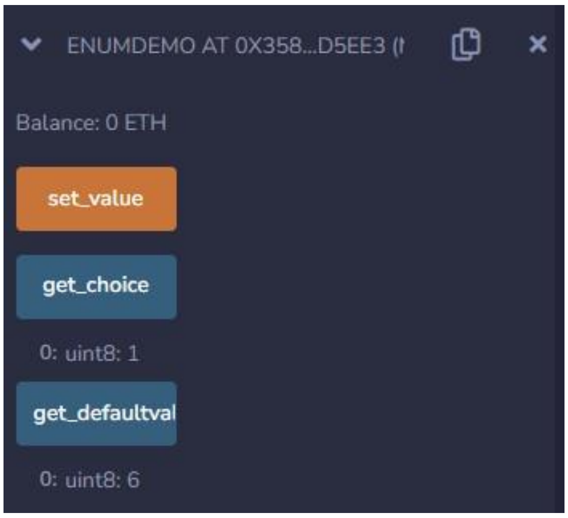


```
DOWHILE AT 0XD91...39138 (ME
Balance: 0 ETH
loop
display
0: uint256[]: 1,2,3,4,5
```

7. Enums

```
pragma solidity ^0.5.0;
contract enumdemo {
     enum week_days {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
}
week_days week;
week_days choice;
week_days constant default_value = week_days.Sunday;
function set_value() public {
        choice = week_days.Tuesday;
}
function get_choice() public view returns (week_days) {
        return choice;
}
function get_defaultvalue() public view returns (week_days) {
        return default_value;
      }
}
```

Output :



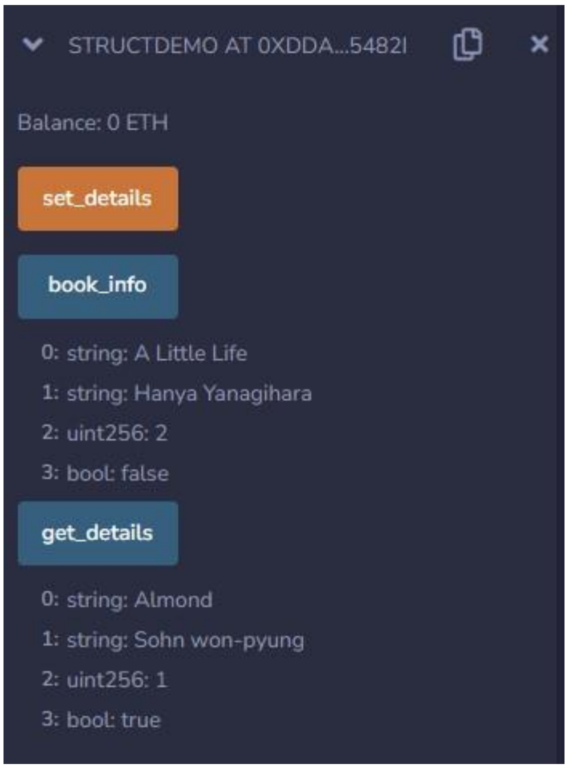
```
0: uint8: 1
0: uint8: 6
```

8. <u>Structs</u>

```
pragma solidity ^0.5.0;
contract structdemo {
    struct Book {
           string name;
           string author;
           uint256 id;
           bool availability;
}
Book book2;
Book book1 = Book("A Little Life", "Hanya Yanagihara", 2, false);
function set_details() public {
           book2 = Book("Almond", "Sohn won-pyung", 1, true);
}
function book_info()
          public
          view
           returns (
                       string memory,
                       string memory,
                       uint256,
                       bool
                     )
{
           return (book1.name, book1.author, book1.id, book1.availability);
}
function get_details()
           public
           view
           returns (
                       string memory, string memory, uint256, bool
)
```

```
    {
        return (book2.name, book2.author, book2.id, book2.availability);
    }
}
```

Output :

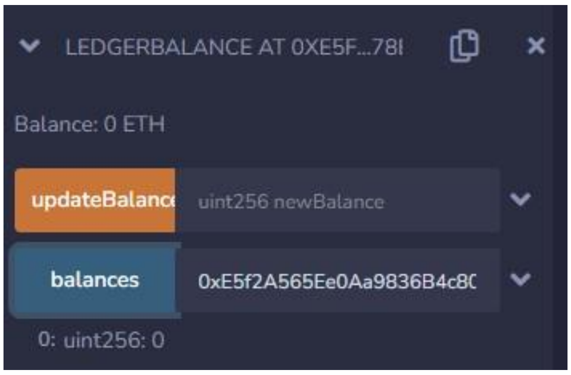


```
STRUCTDEMO AT 0XDDA...5482I

Balance: 0 ETH

set_details

book_info
  0: string: A Little Life
  1: string: Hanya Yanagihara
  2: uint256: 2
  3: bool: false

get_details
  0: string: Almond
  1: string: Sohn won-pyung
  2: uint256: 1
  3: bool: true
```

9. Mappings

```
pragma solidity ^0.5.0;

contract LedgerBalance {
    mapping(address => uint256) public balances;

     function updateBalance(uint256 newBalance) public {
          balances[msg.sender] = newBalance;
     }
}
contract Updater {
     function updateBalance() public returns (uint256) {
         LedgerBalance ledgerBalance = new LedgerBalance();
         return ledgerBalance.balances(address(this));
       }
}
```

Output :



Balance: 0 ETH

0xE5f2A565Ee0Aa9836B4c8C

0: uint256: 0

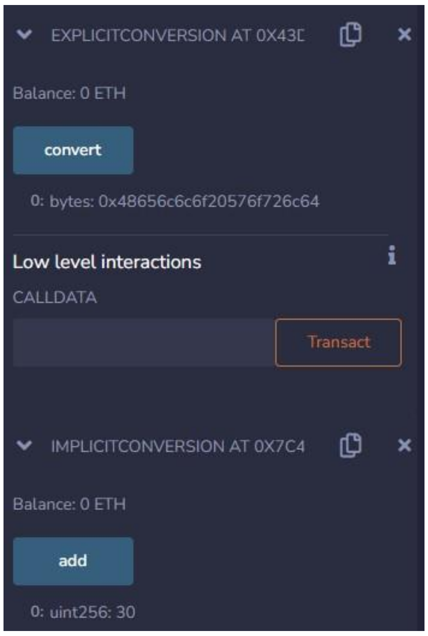
10. Conversions

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ImplicitConversion {
        function add() public pure returns (uint256) {
            uint256 a = 10;
            uint256 b = 20;
            return a + b;
        }
}
contract ExplicitConversion {
            function convert() public pure returns (bytes memory) {
                        string memory str = "Hello World";
                        bytes memory b = bytes(str);
                        return b;
            }
}
```

Steps:

1) Deploy both contracts
2) Open Implicit Conversion and click on add button to sum and display
3) Value
4) Open Explicit Conversion and click on convert button

Output :



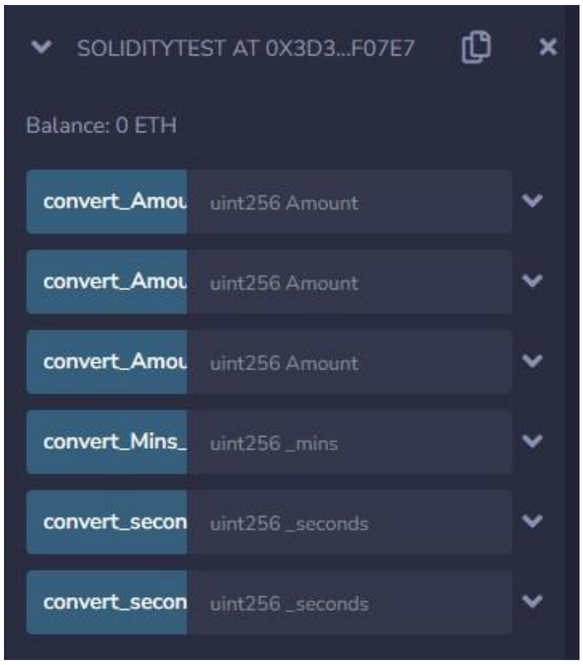
```
0: bytes: 0x48656c6c6f20576f726c64
```

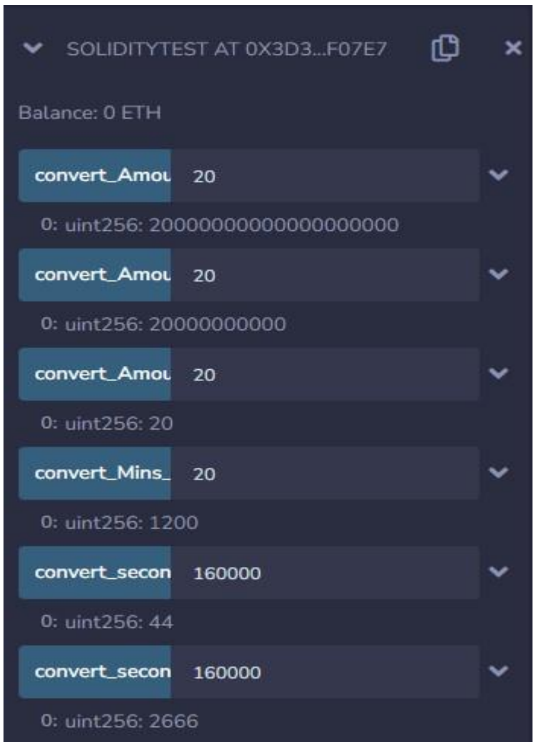
```
0: uint256: 30
```

11. Ether Units

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;
    contract SolidityTest {
        function convert_Amount_to_Wei(uint256 Amount)
                public
                pure
                returns (uint256)
        {
                return Amount * 1 wei;
         }
        function convert_Amount_To_Ether(uint256 Amount)
                public
                pure
                returns (uint256)
        {
                return Amount * 1 ether;
        }
        function convert_Amount_To_Gwei(uint256 Amount)
                public
                pure
                returns (uint256)
        {
                return Amount * 1 gwei;
        }
        function convert_seconds_To_mins(uint256 _seconds)
                public
                pure
                returns (uint256)
        {
                return _seconds / 60;
        }
        function convert_seconds_To_Hours(uint256 _seconds)
                public
                pure
                returns (uint256)
                {
                return _seconds / 3600;
        }
        function convert_Mins_To_Seconds(uint256 _mins)
                public
                pure
                returns (uint256)
                {
                return _mins * 60;
                }
}
```

<u>Output</u> :
Provide values to each function and click on them





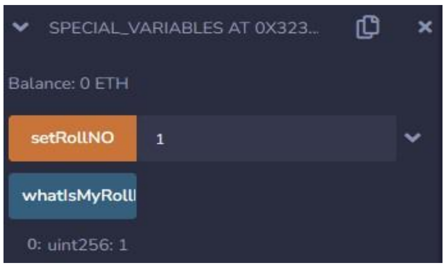
```
Balance: 0 ETH
convert_Amou 20
0: uint256: 20000000000000000000
convert_Amou 20
0: uint256: 20000000000
convert_Amou 20
0: uint256: 20
convert_Mins_ 20
0: uint256: 1200
convert_secon 160000
0: uint256: 44
convert_secon 160000
0: uint256: 2666
```

12. <u>Special Variables</u>

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;
contract Special_Variables {
        mapping(address => uint256) rollNo;
            function setRollNO(uint256 _myNumber) public {
                rollNo[msg.sender] = _myNumber;
            }
      function whatIsMyRollNumber() public view returns (uint256) {
            return rollNo[msg.sender];
            }
}
```

STEPS :
1) Deploy contract Special Variables
2) Input a number for setRollNO function and click on it & whatIsMyRollNumber button

<u>Output</u> :



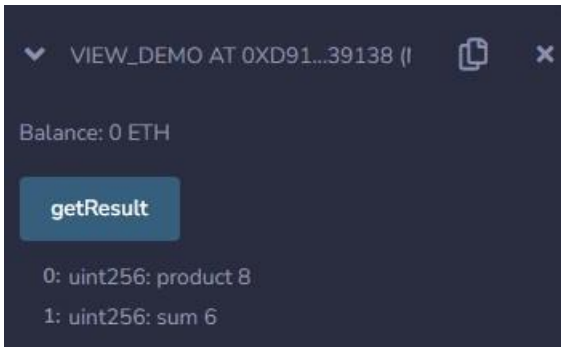
```
Balance: 0 ETH
setRollNO 1
whatIsMyRollI
0: uint256: 1
```

B) Aim : Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions

1. View Functions

```
pragma solidity ^0.8.18;
contract view_demo {
        uint256 num1 = 2;
        uint256 num2 = 4;
        function getResult() public view returns (uint256 product, uint256 sum) {
                product = num1 * num2;
                 sum = num1 + num2;
        }
}
```

Output:

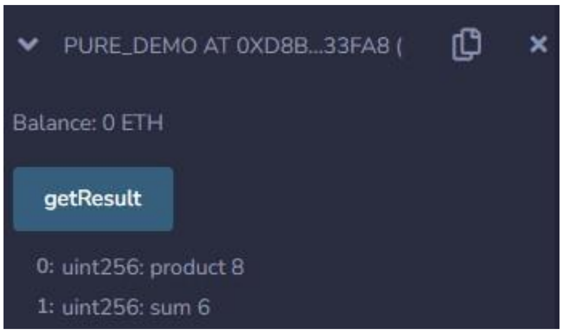


```
VIEW_DEMO AT 0XD91...39138 (I
Balance: 0 ETH
getResult
0: uint256: product 8
1: uint256: sum 6
```

2. Pure Functions

```
pragma solidity ^0.8.18;
contract pure_demo {
        function getResult() public pure returns (uint256 product, uint256 sum) {
                uint256 num1 = 2;
                uint256 num2 = 4;
                product = num1 * num2;
                sum = num1 + num2;
        }
}
```

Output :

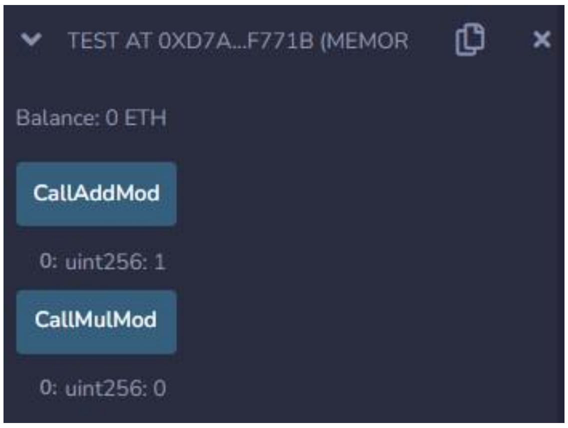


```
PURE_DEMO AT 0XD8B...33FA8 (
Balance: 0 ETH
getResult
0: uint256: product 8
1: uint256: sum 6
```

3. Mathematical Functions

```
pragma solidity ^0.8.18; contract Test{
        function CallAddMod() public pure returns(uint){
                return addmod(7,3,3);
        }
        function CallMulMod() public pure returns(uint){
                return mulmod(7,3,3);
        }
}
```

Output :

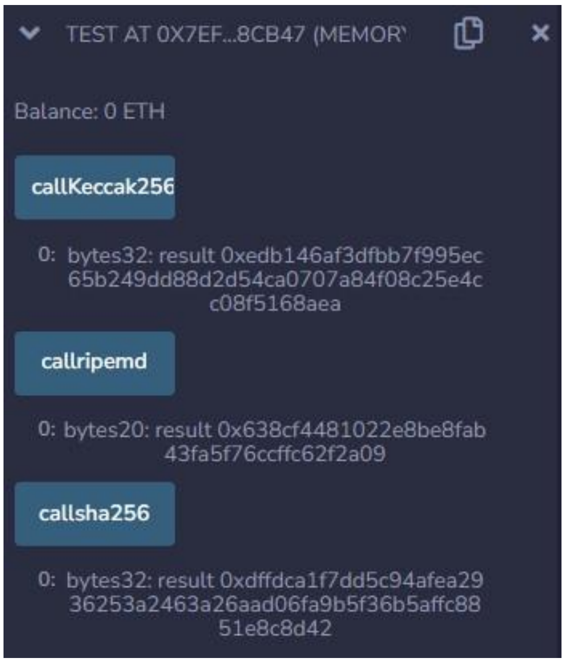


```
Balance: 0 ETH
0: uint256: 1
0: uint256: 0
```

4. Cryptographic Functions

```
pragma solidity ^0.8.18; contract Test{
        function callKeccak256() public pure returns(bytes32 result){
                return keccak256("BLOCKCHAIN");
        }
        function callsha256() public pure returns(bytes32 result){
                return sha256("BLOCKCHAIN");
        }
         function callripemd() public pure returns (bytes20 result){
                return ripemd160("BLOCKCHAIN");
        }
}
```

Output:

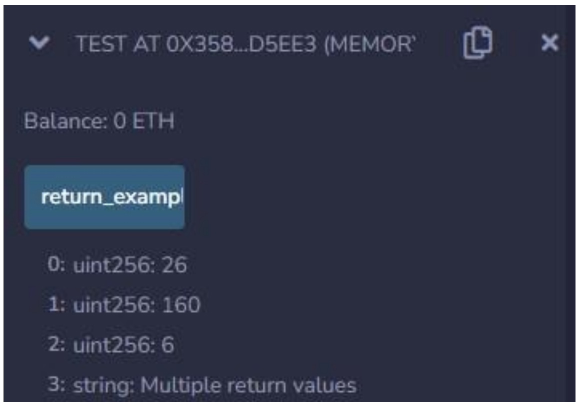


```
Balance: 0 ETH
0: bytes32: result 0xedb146af3dfbb7f995ec
   65b249dd88d2d54ca0707a84f08c25e4c
   c08f5168aea
0: bytes20: result 0x638cf4481022e8be8fab
   43fa5f76ccffc62f2a09
0: bytes32: result 0xdffdca1f7dd5c94afea29
   36253a2463a26aad06fa9b5f36b5affc88
   51e8c8d42
```

5. Functions

```
pragma solidity >=0.4.22 <0.9.0;
contract Test {
        function return_example()
                        public
                        pure
                        returns (
                                    uint256,
                                    uint256,
                                    uint256,
                                        string memory
)
{
            uint256 num1 = 10;
            uint256 num2 = 16;
            uint256 sum = num1 + num2;
            uint256 prod = num1 * num2;
            uint256 diff = num2 - num1;
            string memory message = "Multiple return values";
            return (sum, prod, diff, message);
            }
}
```

Output :

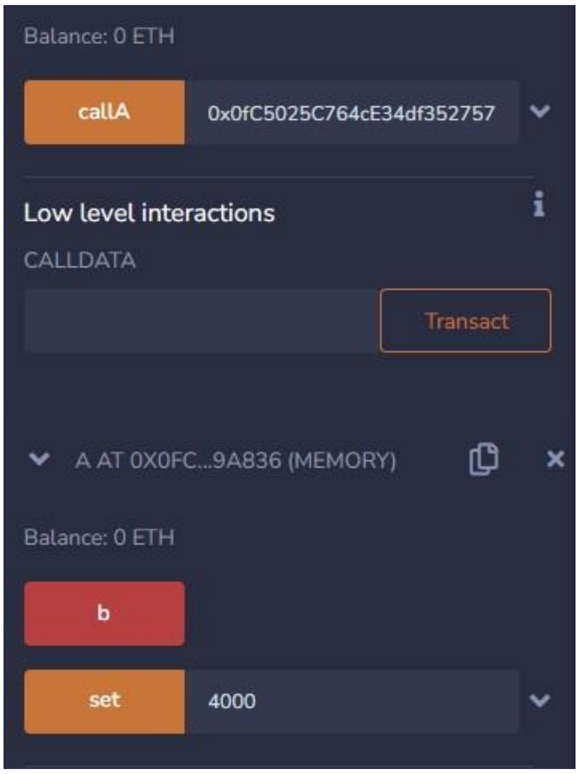


```
Balance: 0 ETH
0: uint256: 26
1: uint256: 160
2: uint256: 6
3: string: Multiple return values
```

6. Fallback Function

```
pragma solidity ^0.5.12;
contract A {
       uint256 n;
      function set(uint256 value) external {
          n = value;
         }
function()   external payable {
       n = 0;
         }
}
contract example {
           function callA(A a) public returns (bool) {
                    (bool success, ) = address(a).call(abi.encodeWithSignature("setter()"));
                    require(success);
                    address payable payableA = address(uint160(address(a)));
                    return (payableA.send(2 ether));
                    }
}
```

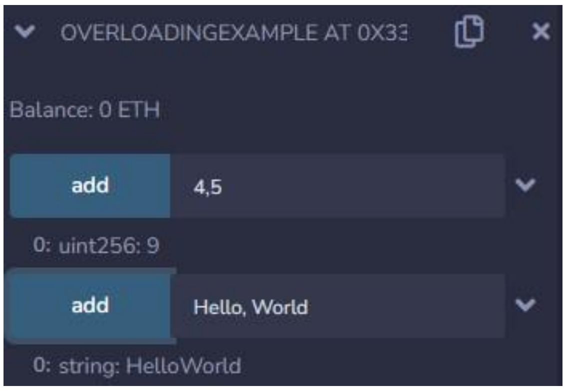
Output :
Provide values to both deployed contracts accordingly(use any address)



7. <u>Function Overloading</u>

```
pragma solidity ^0.8.0;
contract OverloadingExample {
        function add(uint256 a, uint256 b) public pure returns (uint256) {
                return a + b;
        }

        function add(string memory a, string
                memory b)public
                pure
                returns (string memory)
        {
                return string(abi.encodePacked(a, b));
        }
}
```

<u>Output</u> :
Steps:
1) Deploy Overloading Example contract
2) Give integer and string values to both add functions as below

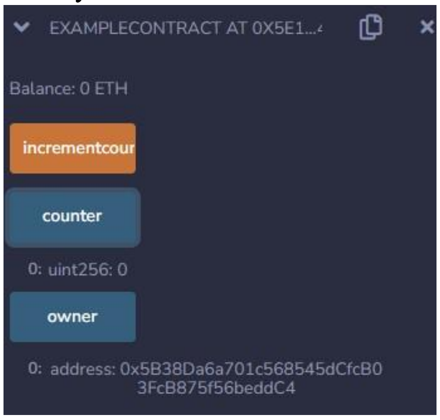


```
Balance: 0 ETH
add 4,5
0: uint256: 9
add Hello, World
0: string: HelloWorld
```

8. Function modifiers

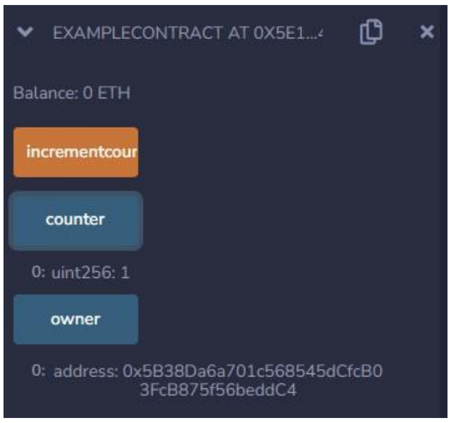
```
pragma solidity ^0.5.0;

contract ExampleContract {
        address public owner =
        0x5B38Da6a701c568545dCfcB03FcB875f56beddC4;uint256 public
        counter;

        modifier onlyowner() {
                require(msg.sender == owner, "Only the contract owner can call");
                _;
        }

        function incrementcounter() public
                onlyowner {counter++;
        }
}
```

<u>Output</u> :
Steps
1) Click on owner button
2) Click on counter button initially it is 0.



```
Balance: 0 ETH
0: uint256: 0
0: address: 0x5B38Da6a701c568545dCfcB0
3FcB875f56beddC4
```

3) Then click on increment counter button and again click on counter button, the counter has been increased.



```
Balance: 0 ETH
0: uint256: 1
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```

PRACTICAL NO : 04

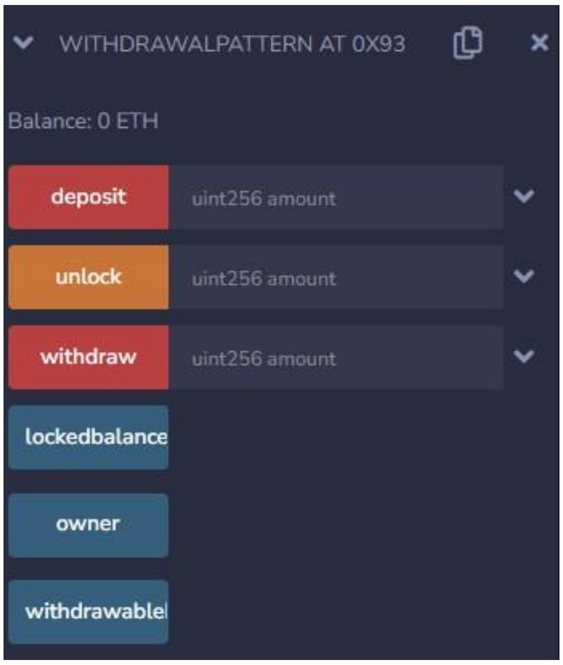
Implement and demonstrate the use of the following in Solidity

A) Aim : Withdrawal Pattern, Restricted Access

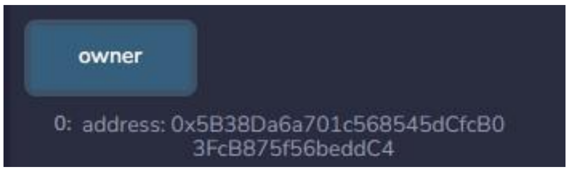
1. Withdrawal Pattern

```
pragma solidity 0.8.18;
contract
        WithdrawalPat
        tern {address
        public owner;
        uint256 public lockedbalance;
        uint256 public withdrawablebalance;

        constructor() {
                owner = msg.sender;
        }

        modifier onlyowner() {
                require(msg.sender == owner, "Only the owner can call this function");
                _;
        }

        function deposit(uint256 amount) public payable {
                require(amount > 0, "Amount must be greater than
                zero");lockedbalance += amount;
        }

        function withdraw(uint256 amount) public payable onlyowner
                {require(
                        amount <= withdrawablebalance,
                        "Insufficient withdrawable balance"
                );
                withdrawablebalance -= amount;
                payable(msg.sender).transfer(amount);
        }

        function unlock(uint256 amount) public onlyowner {
                require(amount <= lockedbalance, "Insufficient locked balance");
                lockedbalance -= amount;
                withdrawablebalance += amount;
        }
}
```

Output :

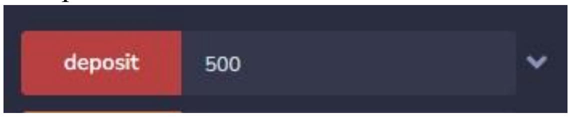


Steps:
Click on owner

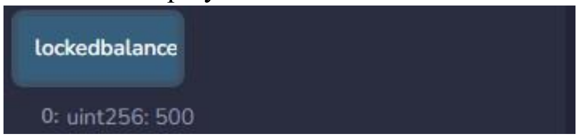


```
0: address: 0x5B38Da6a701c568545dCfcB0
3FcB875f56beddC4
```

Enter an amount and click on deposit

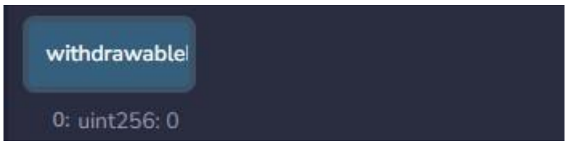


Click on locked balance button to display the locked amount in the account



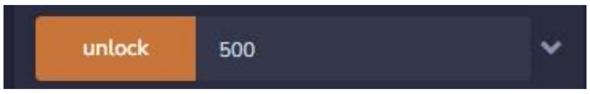
```
0: uint256: 500
```

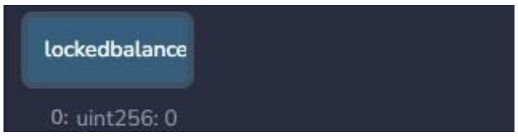
Click on withdrawable balance button

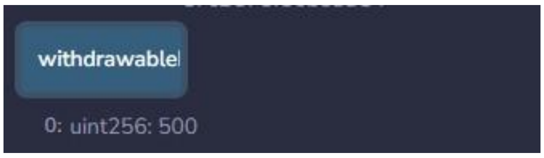


```
0: uint256: 0
```

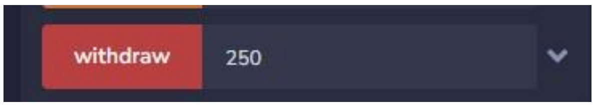
Click on unlock button and enter any amount to transfer amount to withdrawable balance. Check locked balance and withdrawable balance.

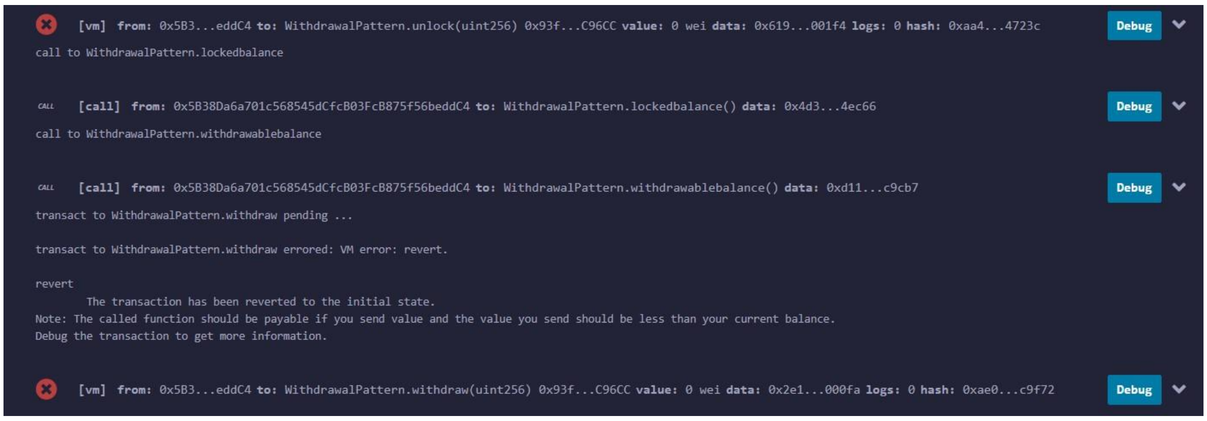






Enter any amount you want to withdraw and click the withdraw button. You should get an error and the transaction should be reverted.





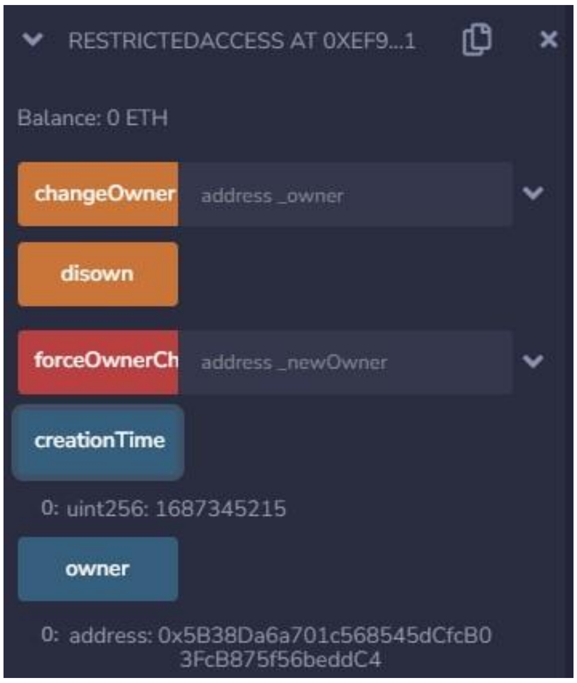
```
[vm] from: 0x5B3...eddC4 to: WithdrawalPattern.unlock(uint256) 0x93f...C96CC value: 0 wei data: 0x619...001f4 logs: 0 hash: 0xaa4...4723c
call to WithdrawalPattern.lockedbalance

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: WithdrawalPattern.lockedbalance() data: 0x4d3...4ec66
call to WithdrawalPattern.withdrawablebalance

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: WithdrawalPattern.withdrawablebalance() data: 0xd11...c9cb7
transact to WithdrawalPattern.withdraw pending ...

transact to WithdrawalPattern.withdraw errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.
Debug the transaction to get more information.

[vm] from: 0x5B3...eddC4 to: WithdrawalPattern.withdraw(uint256) 0x93f...C96CC value: 0 wei data: 0x2e1...000fa logs: 0 hash: 0xae0...c9f72
```

2. Restricted Access

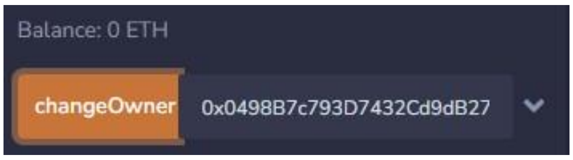
```
pragma solidity ^0.8.18;

contract RestrictedAccess {
        address public owner = msg.sender;
        uint256 public creationTime = block.timestamp;

        modifier onlyBy(address _account) {
                require(msg.sender == _account, "Sender not authorized!");
                _;
        }

        modifier onlyAfter(uint256 _time) {
                require(block.timestamp >= _time, "Function was called too early!");
                _;
        }

        modifier costs(uint256 _amount) {
                require(msg.value >= _amount, "Not enough Ether provided!");
                _;
        }
```

```
    function forceOwnerChange(address
        _newOwner)public
        payable
        costs(200 ether)
    {
        owner = _newOwner;
    }

    function changeOwner(address _owner) public
        onlyBy(owner) {owner = _owner;
    }

    function disown() public onlyBy(owner) onlyAfter(creationTime + 3 weeks) {
        delete owner;
    }
}
```

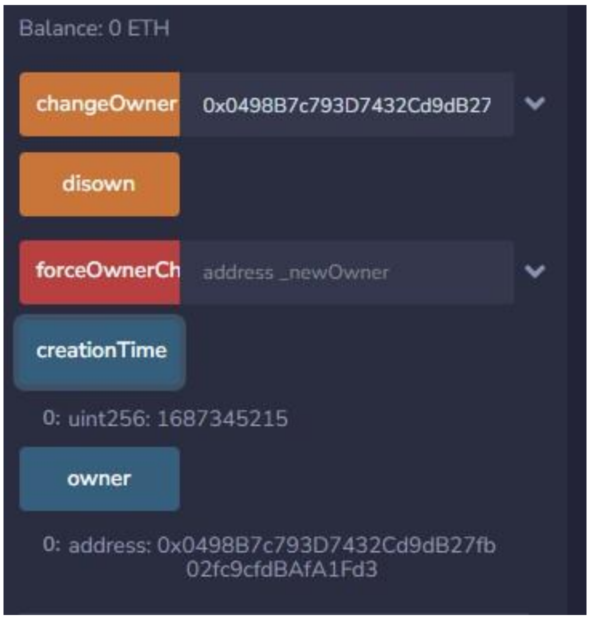
Output :



```
RESTRICTEDACCESS AT 0XEF9...1
Balance: 0 ETH
changeOwner   address _owner
disown
forceOwnerCh   address _newOwner
creationTime
0: uint256: 1687345215
owner
0: address: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
```



```
Balance: 0 ETH
changeOwner   0x0498B7c793D7432Cd9dB27
```



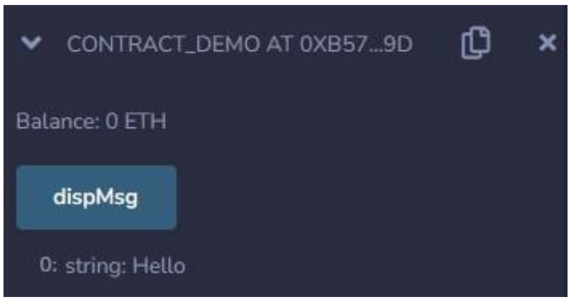
```
0x0498B7c793D7432Cd9dB27
0: uint256: 1687345215
0: address: 0x0498B7c793D7432Cd9dB27fb
            02fc9cfdBAfA1Fd3
```

B) Aim: Contracts, Inheritance, Constructors, Abstract Contracts, Interfaces

1. Contracts

```
pragma solidity ^0.5.0;
contract Contract_demo {
        string message = "Hello";
        function dispMsg() public view returns (string memory)
                {
                return message;
        }
}
```

Output :


CONTRACT_DEMO AT 0XB57...9D

Balance: 0 ETH

0: string: Hello

2. Inheritance

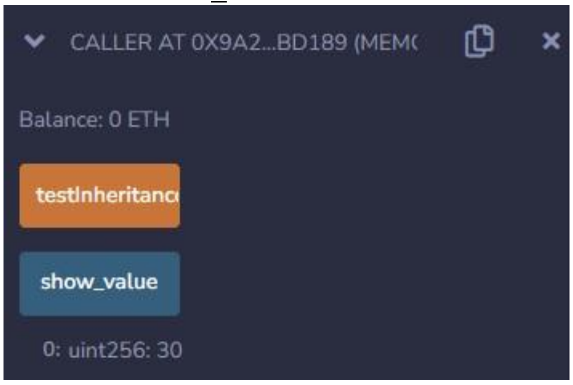
```
pragma solidity >=0.4.22 <0.6.0;
contract Parent {
        uint256 internal sum;
            function setValue() external {
                    uint256 a = 10;
                    uint256 b = 20;
                    sum = a + b;
            }
}
contract child is Parent {
            function getValue() external view returns (uint256) {
                    return sum;
                    }
}

contract caller {
            child cc = new child();
function testInheritance() public returns (uint256) {
            cc.setValue();
            return cc.getValue();
}
        function show_value() public view returns (uint256)
                {return cc.getValue();
}
```

Output :

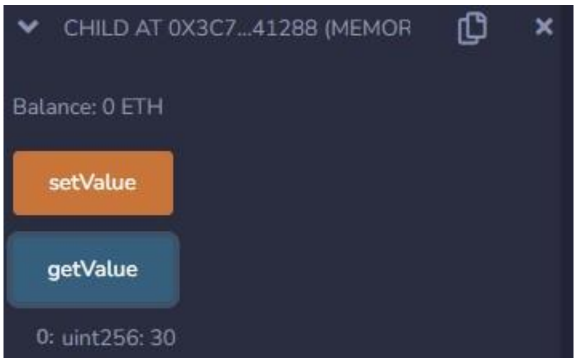
Steps

- Deploy all contracts

Click test Inheritance and then click on show_value to view value



```
0: uint256: 30
```



```
0: uint256: 30
```

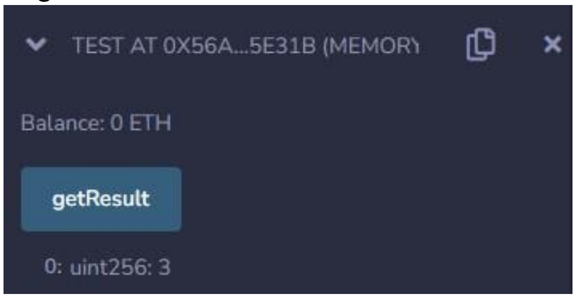
3. <u>Abstract Contract</u>

```
pragma solidity ^0.5.17;
contract Calculator {
        function getResult() external view returns (uint256);
}

contract Test is
        Calculator {
        constructor()
        public {}

        function getResult() external view returns (uint256) {
                uint256 a = 1;
                uint256 b = 2;
                uint256 result
                = a + b;return
                result;
        }
}
```

<u>Output</u> :

Steps

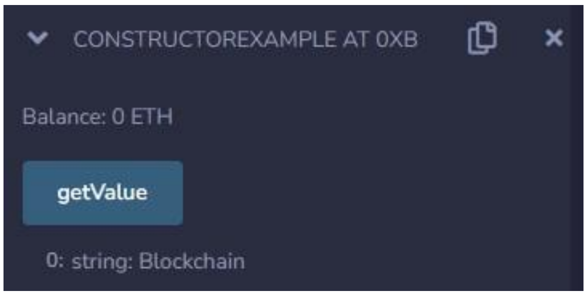
- Deploy test contract
- Click on getResult to get sum of a+b


```
TEST AT 0X56A...5E31B (MEMORY)
Balance: 0 ETH
getResult
0: uint256: 3
```

4. Constructor

```
pragma solidity ^0.5.0;
// Creating a contract
contract
        constructorExa
        mple {string str;

        constructor() public {
                str = "GeeksForGeeks";
        }

        function getValue() public view returns (string memory) {
                return str;
        }
}
```

Output :

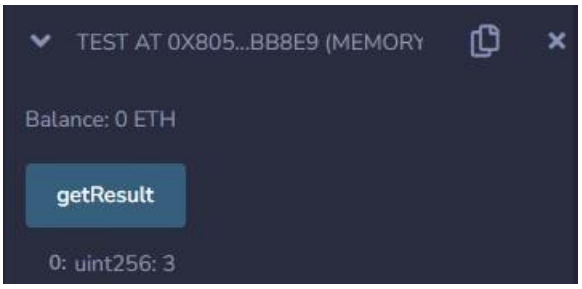

```
CONSTRUCTOREXAMPLE AT 0XB
Balance: 0 ETH
getValue
0: string: Blockchain
```

5. Interfaces

```
pragma solidity ^0.5.0;

interface Calculator {
      function getResult() external view returns(uint);
}
```

```
contract Test is
    Calculator {
    constructor()
    public {}
    function getResult() external view
        returns(uint){uint a = 1;
        uint b = 2;
        uint
        result = a
        + b;return
        result;
    }
}
```

Output :



```
TEST AT 0X805...BB8E9 (MEMORY
Balance: 0 ETH
getResult
0: uint256: 3
```

C) Aim : Libraries, Assembly, Events, Error handling.

1. Libraries
myLib.sol Code

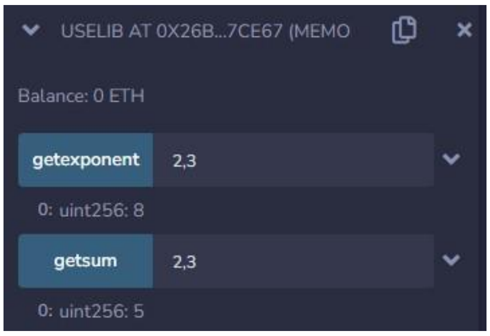
```
pragma solidity >=0.7.0 <0.9.0; library
myMathLib {
   function sum(uint256 a, uint256 b) public pure returns (uint256) {
          return a + b;
   }
   function exponent(uint256 a, uint256 b) public pure returns (uint256) {
          return a**b;
   }
}
```

using_library.sol

```
pragma solidity >=0.7.0 <0.9.0;
import "contracts/myLib.sol";
contract UseLib {
   function getsum(uint256 x, uint256 y) public pure returns (uint256) {
          return myMathLib.sum(x, y);
   }
   function getexponent(uint256 x, uint256 y) public pure returns (uint256) {
          return myMathLib.exponent(x, y);
   }
}
```

Output:
Steps:

- Deploy using_library contract
- Input values to both getexponent and getsum functions as below
- Execute both functions



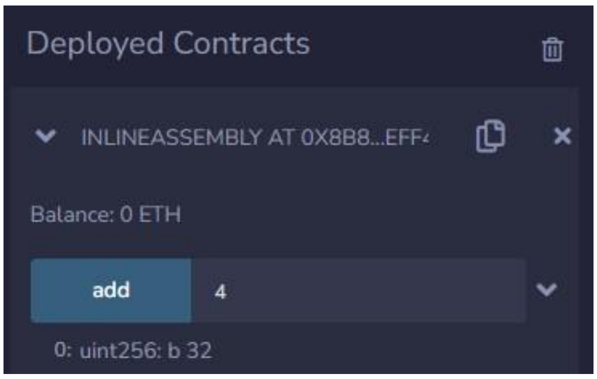
```
USELIB AT 0X26B...7CE67 (MEMO
Balance: 0 ETH
getexponent  2,3
0: uint256: 8
getsum  2,3
0: uint256: 5
```

2. Assembly

```
pragma solidity >=0.4.16 <0.9.0;
contract InlineAssembly {
// Defining function
            function add(uint256 a) public view returns (uint256 b) {
                     assembly {
```

```
                    let c := add(a, 16) mstore(0x80, c)
                    {
                            let d := add(sload(c), 12) b := d
                    }

                    b := add(b, c)
            }
        }
}
```

Output :



Deployed Contracts

INLINEASSEMBLY AT 0X8B8...EFF4

Balance: 0 ETH

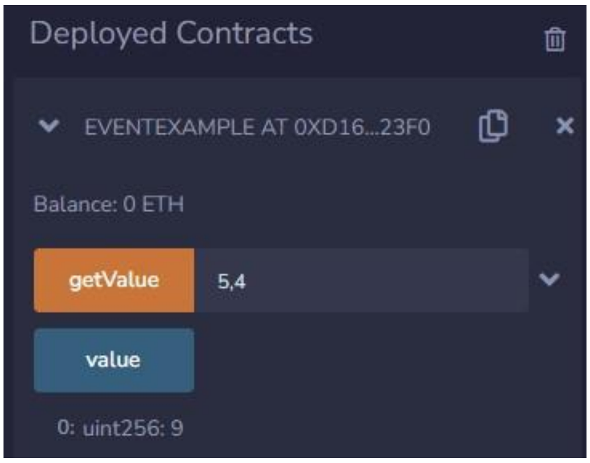
add 4

0: uint256: b 32

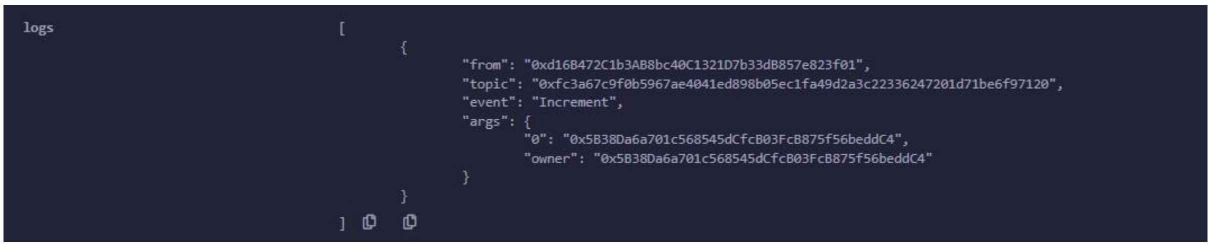
3. <u>Events</u>

```
pragma solidity ^0.5.0;

// Creating a contract contract eventExample {
// Declaring state variables uint256 public value = 0;

// Declaring an event
event Increment(address owner);

// Defining a function for logging event
function getValue(uint256 _a, uint256 _b) public { emit Increment(msg.sender);
value = _a + _b;
}
}
```

<u>Output</u> :



```
0: uint256: 9
```



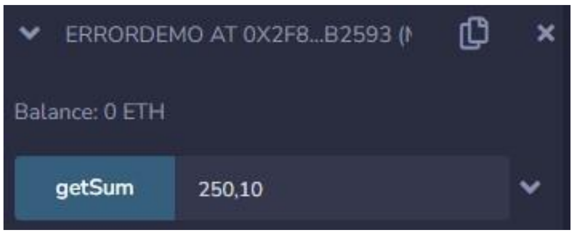
```
logs                    [
                                {
                                        "from": "0xd16B472C1b3AB8bc40C1321D7b33dB857e823f01",
                                        "topic": "0xfc3a67c9f0b5967ae4041ed898b05ec1fa49d2a3c22336247201d71be6f97120",
                                        "event": "Increment",
                                        "args": {
                                                "0": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4",
                                                "owner": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                                        }
                                }
                        ]
```

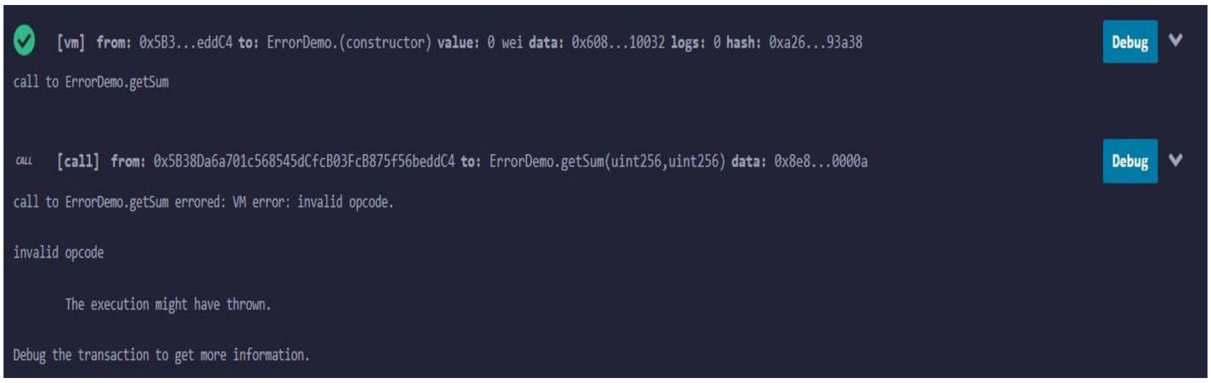
4. <u>Error Handling</u>

```
solidity ^0.5.17;
 contract ErrorDemo {
          function getSum(uint256 a, uint256 b) public pure returns (uint256) {
                    uint256 sum = a + b;
                    assert(sum<255);
                    return sum;
}
```

<u>Output</u> :
Steps:

- Provide some values and press on getSum
- Check terminal panel





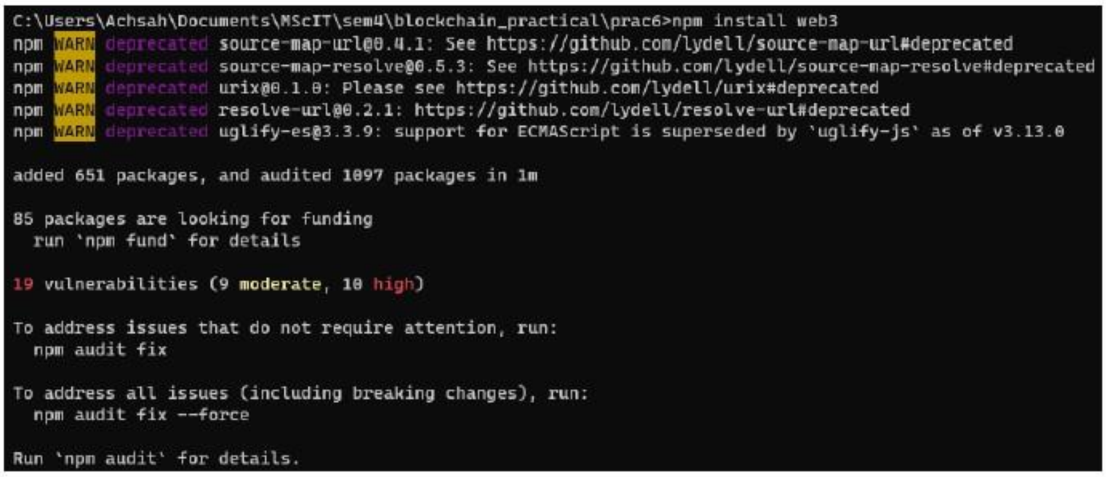
```
[vm] from: 0x5B3...eddC4 to: ErrorDemo.(constructor) value: 0 wei data: 0x608...10032 logs: 0 hash: 0xa26...93a38
call to ErrorDemo.getSum

CALL [call] from: 0x5B38Da6a701c568545dCfcB03FcB875f56beddC4 to: ErrorDemo.getSum(uint256,uint256) data: 0x8e8...0000a
call to ErrorDemo.getSum errored: VM error: invalid opcode.

invalid opcode

        The execution might have thrown.

Debug the transaction to get more information.
```

PRACTICAL NO : 05

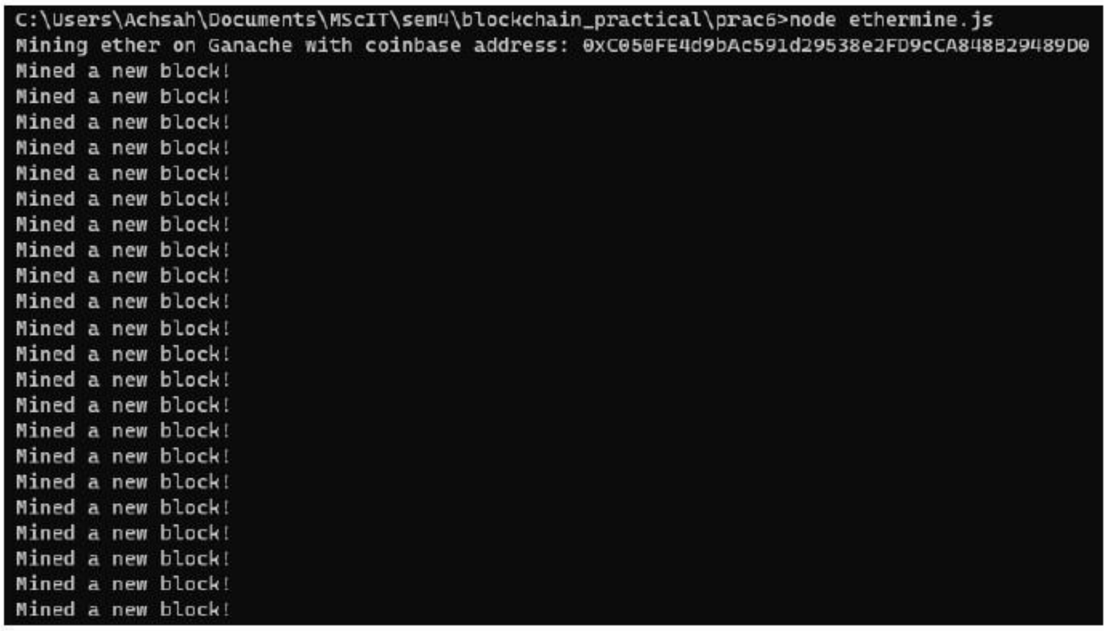
Write a program to demonstrate mining of Ether.

```
const Web3 = require('web3');
const web3 = new Web3(new Web3.providers.HttpProvider('http: /127.0.0.1:7545')); / Replace
with your Ganache HTTP provider
async function mine() {
	const accounts = await web3.eth.getAccounts();
	const coinbaseacc1 = accounts[0];
	const coinbaseacc2 = accounts[1];
	console.log(`Mining ether on Ganache with coinbase address:
${coinbaseacc1}`);
	while (true) {
		try {
			await web3.eth.sendTransaction({
				from: coinbaseacc1,
				to: coinbaseacc2,
				value: 50,
			});
			console.log(`Mined a new block!`);
		} catch (err) {
			console.error(err);
		}
	}
}
mine();
```

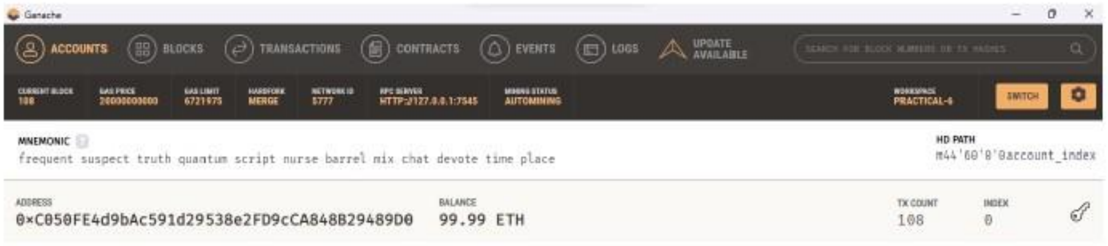
Output :



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac6>npm install web3
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
npm WARN deprecated source-map-resolve@0.5.3: See https://github.com/lydell/source-map-resolve#deprecated
npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
npm WARN deprecated uglify-es@3.3.9: support for ECMAScript is superseded by `uglify-js` as of v3.13.0

added 651 packages, and audited 1097 packages in 1m

85 packages are looking for funding
  run `npm fund` for details

19 vulnerabilities (9 moderate, 10 high)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac6>node ethermine.js
Mining ether on Ganache with coinbase address: 0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
Mined a new block!
```



CURRENT BLOCK 108 | GAS PRICE 20000000000 | GAS LIMIT 6721975 | HARDFORK MERGE | NETWORK ID 5777 | RPC SERVER HTTP://127.0.0.1:7545 | MINING STATUS AUTOMINING | WORKSPACE PRACTICAL-6

MNEMONIC
frequent suspect truth quantum script nurse barrel mix chat devote time place

HD PATH
m44'60'0'0account_index

ADDRESS 0×C050FE4d9bAc591d29538e2FD9cCA848B29489D0 | BALANCE 99.99 ETH | TX COUNT 108 | INDEX 0

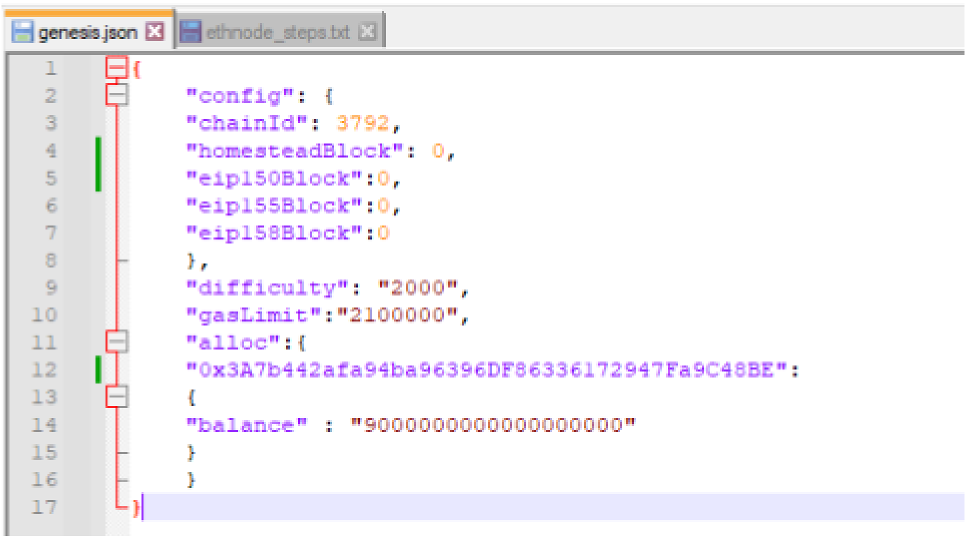
PRACTICAL NO : 06

Demonstrate the running of the blockchain node.

Step 1-> Create a folder named ethermine and a JSON file named genesis.json and write the following lines in it.

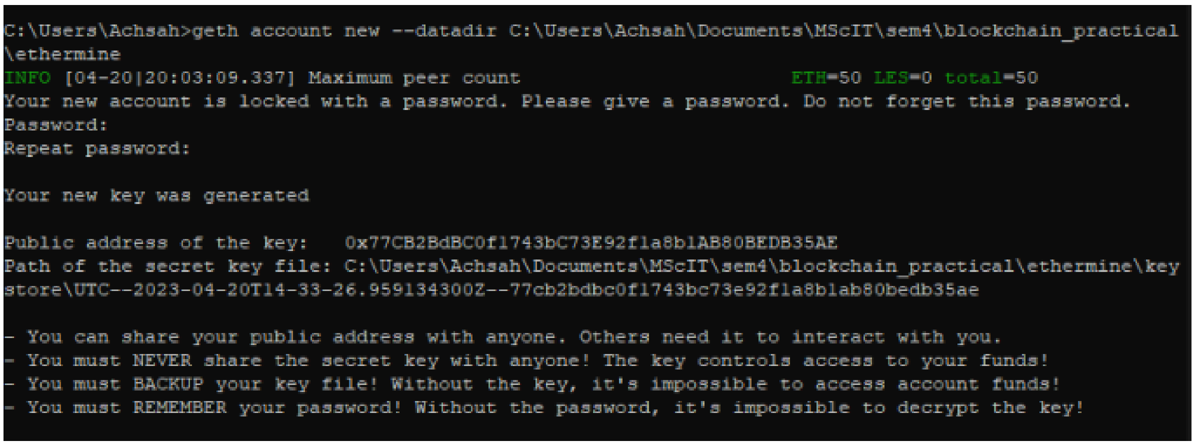
```
{
        "config": {
                "chainId": 3792,
                "homesteadBlock": 0,
                "eip150Block": 0,
                "eip155Block": 0,
                "eip158Block": 0
        },
        "difficulty": "2000",
        "gasLimit": "2100000",
        "alloc": {
                "0x0b6C4c81f58B8d692A7B46AD1e16a1147c25299F": {
                        "balance": "9000000000000000000"
                }
        }
}
```

Output :


genesis.json ethnode_steps.txt

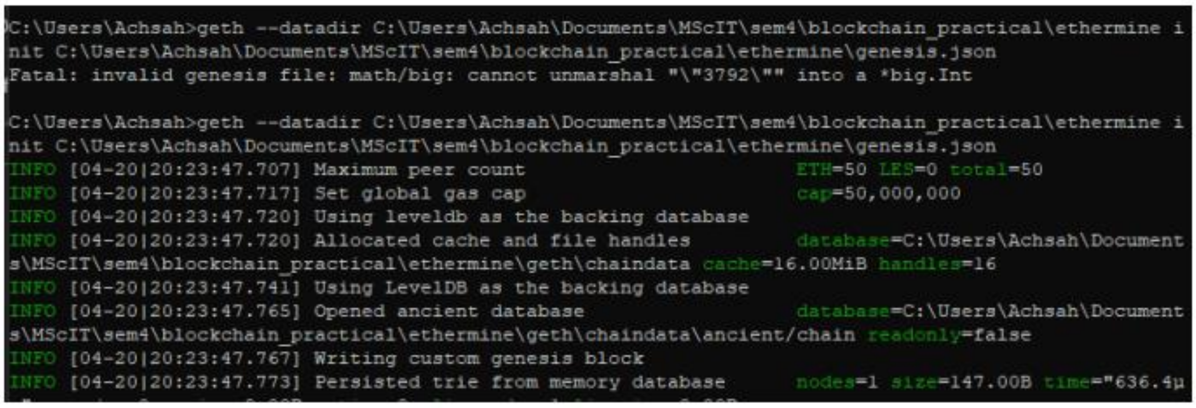
```
{
    "config": {
    "chainId": 3792,
    "homesteadBlock": 0,
    "eip150Block":0,
    "eip155Block":0,
    "eip158Block":0
    },
    "difficulty": "2000",
    "gasLimit":"2100000",
    "alloc":{
    "0x3A7b442afa94ba96396DF86336172947Fa9C48BE":
    {
    "balance" : "9000000000000000000"
    }
    }
}
```

Step 2-> Run command geth account new –datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine testnet-blockchain



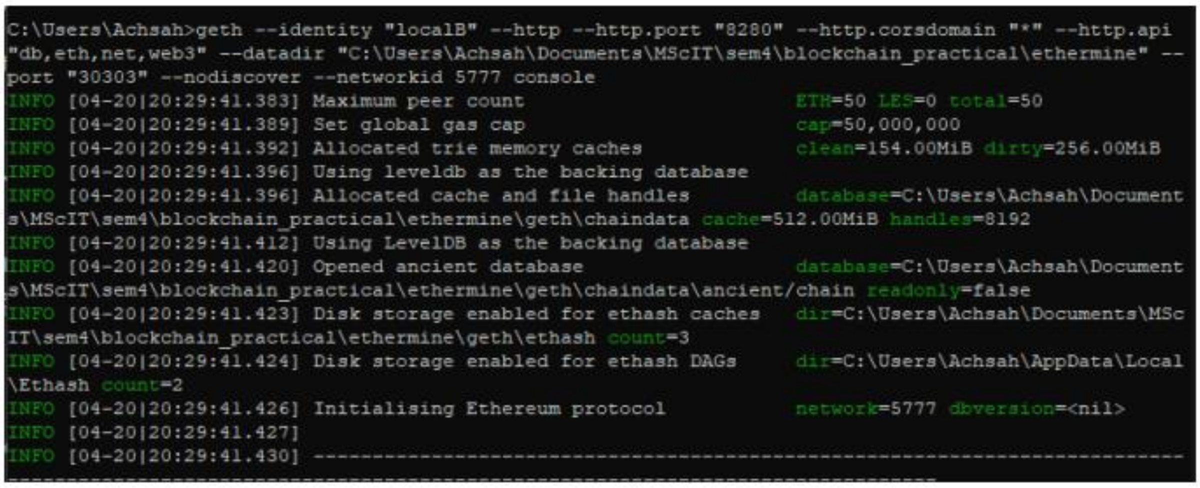
```
C:\Users\Achsah>geth account new --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical
\ethermine
INFO [04-20|20:03:09.337] Maximum peer count                       ETH=50 LES=0 total=50
Your new account is locked with a password. Please give a password. Do not forget this password.
Password:
Repeat password:

Your new key was generated

Public address of the key:   0x77CB2BdBC0f1743bC73E92f1a8b1AB80BEDB35AE
Path of the secret key file: C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\key
store\UTC--2023-04-20T14-33-26.959134300Z--77cb2bdbc0f1743bc73e92f1a8b1ab80bedb35ae

- You can share your public address with anyone. Others need it to interact with you.
- You must NEVER share the secret key with anyone! The key controls access to your funds!
- You must BACKUP your key file! Without the key, it's impossible to access account funds!
- You must REMEMBER your password! Without the password, it's impossible to decrypt the key!
```

Step 3-> Run command geth account new --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine



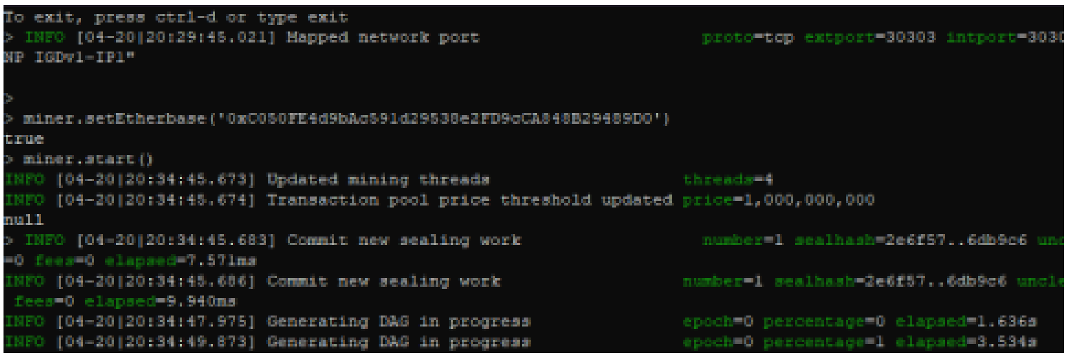
```
C:\Users\Achsah>geth --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine i
nit C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\genesis.json
Fatal: invalid genesis file: math/big: cannot unmarshal "\"3792\"" into a *big.Int

C:\Users\Achsah>geth --datadir C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine i
nit C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine\genesis.json
INFO [04-20|20:23:47.707] Maximum peer count                       ETH=50 LES=0 total=50
INFO [04-20|20:23:47.717] Set global gas cap                       cap=50,000,000
INFO [04-20|20:23:47.720] Using leveldb as the backing database
INFO [04-20|20:23:47.720] Allocated cache and file handles         database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata cache=16.00MiB handles=16
INFO [04-20|20:23:47.741] Using LevelDB as the backing database
INFO [04-20|20:23:47.765] Opened ancient database                  database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata\ancient/chain readonly=false
INFO [04-20|20:23:47.767] Writing custom genesis block
INFO [04-20|20:23:47.773] Persisted trie from memory database      nodes=1 size=147.00B time="636.4µ
```

Step 4-> Run command geth --identity "localB" --http --http.port "8280"--http.corsdomain "*" --http.api "db,eth,net,web3" --datadir "C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine" --port "30303" --nodiscover --networkid 5777 console. This command will enable geth console.



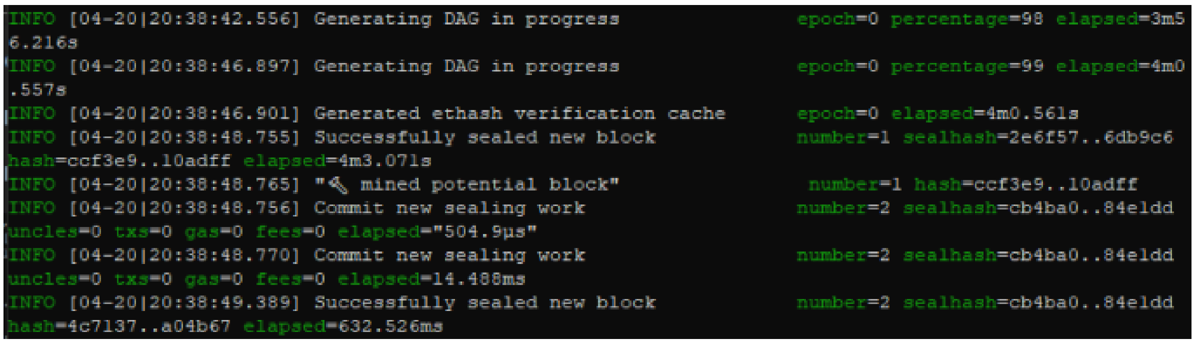
```
C:\Users\Achsah>geth --identity "localB" --http --http.port "8280" --http.corsdomain "*" --http.api
"db,eth,net,web3" --datadir "C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\ethermine" --
port "30303" --nodiscover --networkid 5777 console
INFO [04-20|20:29:41.383] Maximum peer count                       ETH=50 LES=0 total=50
INFO [04-20|20:29:41.389] Set global gas cap                       cap=50,000,000
INFO [04-20|20:29:41.392] Allocated trie memory caches             clean=154.00MiB dirty=256.00MiB
INFO [04-20|20:29:41.396] Using leveldb as the backing database
INFO [04-20|20:29:41.396] Allocated cache and file handles         database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata cache=512.00MiB handles=8192
INFO [04-20|20:29:41.412] Using LevelDB as the backing database
INFO [04-20|20:29:41.420] Opened ancient database                  database=C:\Users\Achsah\Document
s\MScIT\sem4\blockchain_practical\ethermine\geth\chaindata\ancient/chain readonly=false
INFO [04-20|20:29:41.423] Disk storage enabled for ethash caches   dir=C:\Users\Achsah\Documents\MSc
IT\sem4\blockchain_practical\ethermine\geth\ethash count=3
INFO [04-20|20:29:41.424] Disk storage enabled for ethash DAGs     dir=C:\Users\Achsah\AppData\Local
\Ethash count=2
INFO [04-20|20:29:41.426] Initialising Ethereum protocol           network=5777 dbversion=<nil>
INFO [04-20|20:29:41.427]
INFO [04-20|20:29:41.430] ---------------------------------------------------------------------------
------------------------------------------------------------------------
```

Step 5-> Run the command miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0’) in the geth console
Step 6-> Run the command miner.start() to start mining



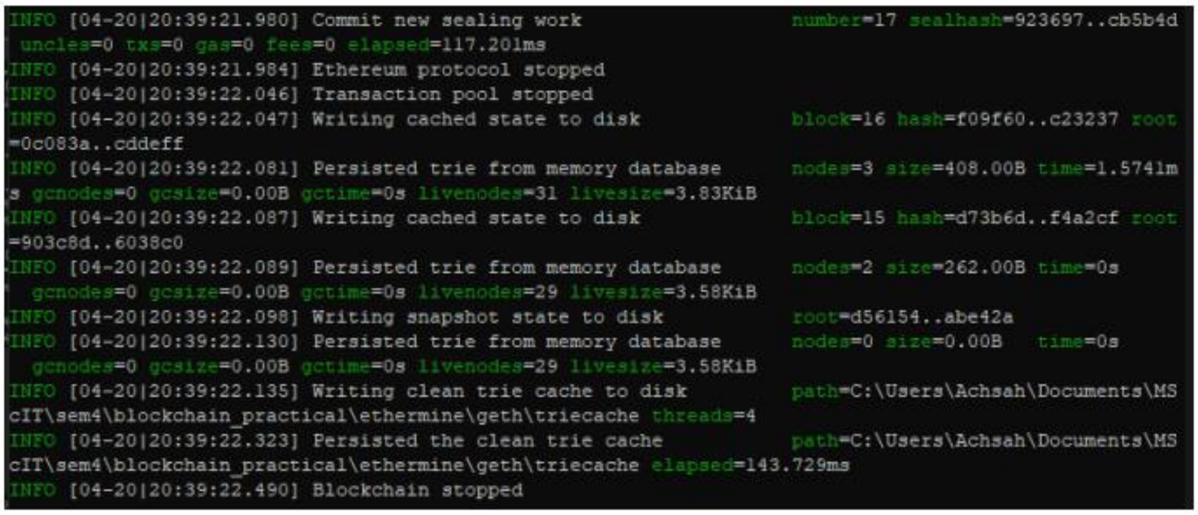
```
To exit, press ctrl-d or type exit
> INFO [04-20|20:29:45.021] Mapped network port                      proto=tcp extport=30303 intport=3030
NP IGDv1-IP1"

>
> miner.setEtherbase('0xC050FE4d9bAc591d29538e2FD9cCA848B29489D0')
true
> miner.start()
INFO [04-20|20:34:45.673] Updated mining threads                   threads=4
INFO [04-20|20:34:45.674] Transaction pool price threshold updated price=1,000,000,000
null
> INFO [04-20|20:34:45.683] Commit new sealing work                  number=1 sealhash=2e6f57..6db9c6 unc
=0 fees=0 elapsed=7.571ms
INFO [04-20|20:34:45.686] Commit new sealing work                  number=1 sealhash=2e6f57..6db9c6 uncle
 fees=0 elapsed=9.940ms
INFO [04-20|20:34:47.975] Generating DAG in progress               epoch=0 percentage=0 elapsed=1.636s
INFO [04-20|20:34:49.873] Generating DAG in progress               epoch=0 percentage=1 elapsed=3.534s
```

Step 7-> Below screenshots are the mining processes running on your local machine.



```
INFO [04-20|20:38:42.556] Generating DAG in progress               epoch=0 percentage=98 elapsed=3m5
6.216s
INFO [04-20|20:38:46.897] Generating DAG in progress               epoch=0 percentage=99 elapsed=4m0
.557s
INFO [04-20|20:38:46.901] Generated ethash verification cache      epoch=0 elapsed=4m0.561s
INFO [04-20|20:38:48.755] Successfully sealed new block            number=1 sealhash=2e6f57..6db9c6
hash=ccf3e9..10adff elapsed=4m3.071s
INFO [04-20|20:38:48.765] "🔨 mined potential block"                number=1 hash=ccf3e9..10adff
INFO [04-20|20:38:48.756] Commit new sealing work                  number=2 sealhash=cb4ba0..84e1dd
uncles=0 txs=0 gas=0 fees=0 elapsed="504.9µs"
INFO [04-20|20:38:48.770] Commit new sealing work                  number=2 sealhash=cb4ba0..84e1dd
uncles=0 txs=0 gas=0 fees=0 elapsed=14.488ms
INFO [04-20|20:38:49.389] Successfully sealed new block            number=2 sealhash=cb4ba0..84e1dd
hash=4c7137..a04b67 elapsed=632.526ms
```

Step 8-> To stop the mining press Ctrl+D



```
INFO [04-20|20:39:21.980] Commit new sealing work                  number=17 sealhash=923697..cb5b4d
 uncles=0 txs=0 gas=0 fees=0 elapsed=117.201ms
INFO [04-20|20:39:21.984] Ethereum protocol stopped
INFO [04-20|20:39:22.046] Transaction pool stopped
INFO [04-20|20:39:22.047] Writing cached state to disk             block=16 hash=f09f60..c23237 root
=0c083a..cddeff
INFO [04-20|20:39:22.081] Persisted trie from memory database      nodes=3 size=408.00B time=1.5741m
s gcnodes=0 gcsize=0.00B gctime=0s livenodes=31 livesize=3.83KiB
INFO [04-20|20:39:22.087] Writing cached state to disk             block=15 hash=d73b6d..f4a2cf root
=903c8d..6038c0
INFO [04-20|20:39:22.089] Persisted trie from memory database      nodes=2 size=262.00B time=0s
  gcnodes=0 gcsize=0.00B gctime=0s livenodes=29 livesize=3.58KiB
INFO [04-20|20:39:22.098] Writing snapshot state to disk           root=d56154..abe42a
INFO [04-20|20:39:22.130] Persisted trie from memory database      nodes=0 size=0.00B   time=0s
  gcnodes=0 gcsize=0.00B gctime=0s livenodes=29 livesize=3.58KiB
INFO [04-20|20:39:22.135] Writing clean trie cache to disk         path=C:\Users\Achsah\Documents\MS
cIT\sem4\blockchain_practical\ethermine\geth\triecache threads=4
INFO [04-20|20:39:22.323] Persisted the clean trie cache           path=C:\Users\Achsah\Documents\MS
cIT\sem4\blockchain_practical\ethermine\geth\triecache elapsed=143.729ms
INFO [04-20|20:39:22.490] Blockchain stopped
```

PRACTICAL NO : 07

Create your own blockchain and demonstrate its use.

Create a javascript folder with the following code in any folder of your choice.
JavaScript Code

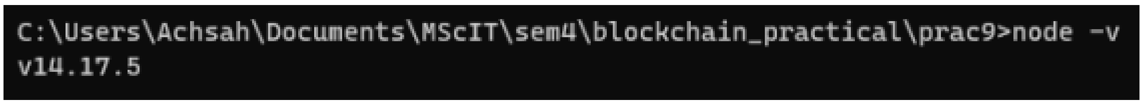
```
const SHA256 = require("crypto-js/sha256");
class Block {
	constructor(index, timestamp, data, previousHash = "") {
		this.index = index;
		this.timestamp = timestamp;
		this.data = data;
		this.previousHash = previousHash;
		this.hash = this.calculateHash();
	}
	calculateHash() {
		return SHA256(
			this.index +
			this.previousHash +
			this.timestamp +
			JSON.stringify(this.data)
		).toString();
	}
}

class Blockchain {
	constructor() {
		this.chain = [this.createGenesisBlock()];
	}
	createGenesisBlock() {
		return new Block(0, "21/04/2023", "Genesis Block", "0");
	}
	getLatestBlock() {
		return this.chain[this.chain.length - 1];
	}
	addBlock(newBlock) {
		newBlock.previousHash = this.getLatestBlock().hash;
		newBlock.hash = newBlock.calculateHash();
		this.chain.push(newBlock);
	}
	isChainValid() {
		for (let i = 1; i < this.chain.length; i +) {
			const currentBlock = this.chain[i];
			const previousBlock = this.chain[i - 1];
			if (currentBlock.hash = currentBlock.calculateHash()) {
```

```
                        return false;
                    }
                    if (currentBlock.previousHash = previousBlock.hash) {
                        return false;
                    }
            }
            return true;
    }
}
```

/Blockchain Implementation

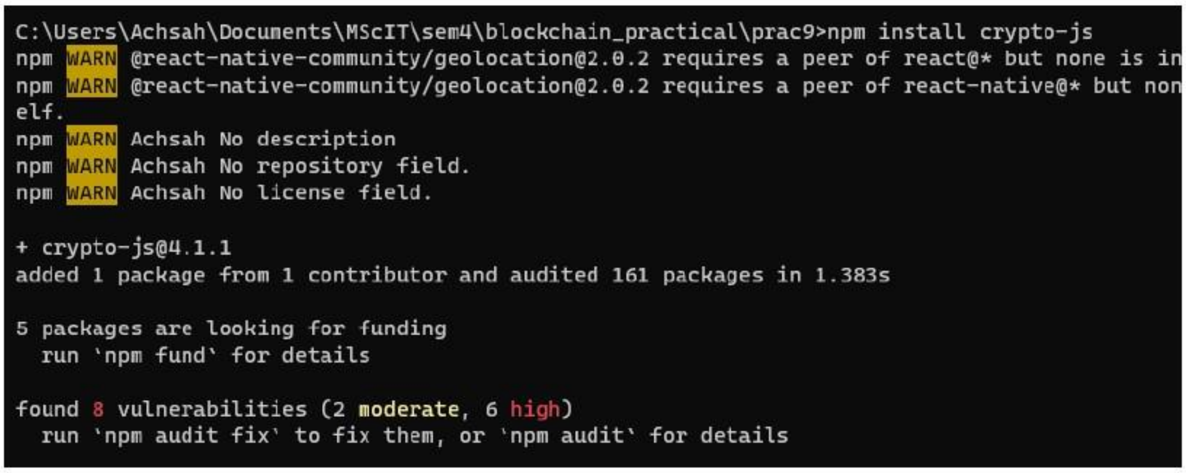
```
let myCoin = new Blockchain();
myCoin.addBlock(new Block(1, "22/04/2023", { amount: 4 }));
myCoin.addBlock(new Block(2, "22/04/2023", { amount: 8 }));
//console.log('Is blockchain valid? ' + myCoin.isChainValid());
console.log(JSON.stringify(myCoin, null, 4));
```

<u>Output</u> :
Flow of execution
Step 1-> Make sure you have installed nodejs in your system



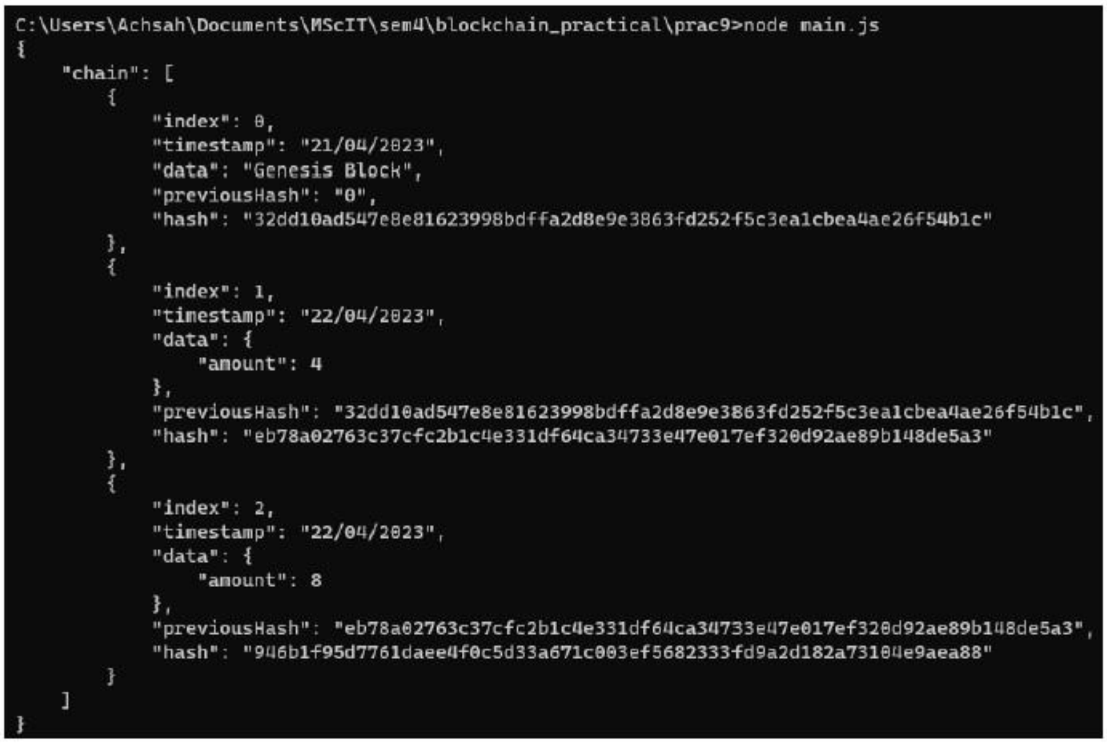
```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>node -v
v14.17.5
```

Step 2-> We need crypto –js node module to make our own blockchain. So install it as following



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>npm install crypto-js
npm WARN @react-native-community/geolocation@2.0.2 requires a peer of react@* but none is in
npm WARN @react-native-community/geolocation@2.0.2 requires a peer of react-native@* but non
elf.
npm WARN Achsah No description
npm WARN Achsah No repository field.
npm WARN Achsah No license field.

+ crypto-js@4.1.1
added 1 package from 1 contributor and audited 161 packages in 1.383s

5 packages are looking for funding
  run `npm fund` for details

found 8 vulnerabilities (2 moderate, 6 high)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Step 3-> Run the above code in command line using command: node main.js



```
C:\Users\Achsah\Documents\MScIT\sem4\blockchain_practical\prac9>node main.js
{
    "chain": [
        {
            "index": 0,
            "timestamp": "21/04/2023",
            "data": "Genesis Block",
            "previousHash": "0",
            "hash": "32dd10ad547e8e81623998bdffa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c"
        },
        {
            "index": 1,
            "timestamp": "22/04/2023",
            "data": {
                "amount": 4
            },
            "previousHash": "32dd10ad547e8e81623998bdffa2d8e9e3863fd252f5c3ea1cbea4ae26f54b1c",
            "hash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3"
        },
        {
            "index": 2,
            "timestamp": "22/04/2023",
            "data": {
                "amount": 8
            },
            "previousHash": "eb78a02763c37cfc2b1c4e331df64ca34733e47e017ef320d92ae89b148de5a3",
            "hash": "946b1f95d7761daee4f0c5d33a671c003ef5682333fd9a2d182a73104e9aea88"
        }
    ]
}
```